University of Zurich[UZH]

Department of Informatics

Towards Effective Support for Data Mining using Intelligent Discovery Assistance

A dissertation submitted to the

FACULTY OF ECONOMICS, BUSINESS ADMINISTRATION AND INFORMATION TECHNOLOGY OF THE UNIVERSITY OF ZURICH

for the degree of

DOCTOR OF SCIENCE

by

FLOAREA SERBAN
from Romania

accepted on the recommendation of

Prof. Abraham Bernstein, Ph.D.
Prof. Dr. Michèle Sebag
Prof. Dr. Filip Zelezny

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich herewith permits the publication of the aforementioned dissertation without expressing any opinion on the views contained therein.

Zurich, April 24, 2013

Head of the Ph.D. committee for informatics: Prof. Abraham Bernstein, Ph.D.

Acknowledgements

My time at the University of Zurich was influenced by many people to whom I am extremely grateful. First, I would like to express my deep gratitude to my supervisor Abraham Bernstein for his support and advice during my Phd. I appreciate that he allowed me to explore various research areas and guided my path. I remember that several times I was out of ideas and could not find a solution, but after talking to him I was back on the right track. Thank you, Avi, for this!

I could not have done this work without Juk's guidance. He always shared his experience from Data Mining and helped me see the tricky hidden problems. I learned so much from him and also enjoyed all the conferences that we attended together. I am really grateful for all his help and valuable and constructive suggestions.

Special thanks to my external supervisors Michèle Sebag and Filip Zelezny. I am grateful to Michèle for the new ideas and feedback that she provided. These opened new paths to my research.

The atmosphere at IFI was great and I learned a lot about research in general. I enjoyed being there and I wish to especially thank my former office mates: Jonas, Jayalath, Ping, Philip, Thomas and Patrick. I had a great time and I appreciate the fruitful conversations that we had. Philip always challenged me with never ending questions and ideas. Tom introduced me to the German culture and language – Danke schön! I would like to thank all the other DDIS members for the great coffee breaks, discussions and time that we shared together (Kathi, Amancio, Doro, Cosmin, Iris, Ela, Khoa).

My appreciation also goes to my teachers and professors from Romania. They inspired me and contributed to my path towards research in science.

I would not have written this thesis without the constant support of my family. They encouraged me and understood me in very difficult moments. Finally, I would like to thank Ionut for his patience, encouragements and help throughout my thesis.

Many thanks to all
Flori

Zürich, April 2013

Abstract

Living in the Information Age, Knowledge Discovery for Large Databases (KDD) plays a crucial role in analyzing the ever increasing amount of data. People need access to efficient data analysis tools that simplify the whole process, but also provide good results. One of the largest impediments of such tools is the lack of support for the whole process. The large number of methods available aggravates this problem even more. While research has shown that it is possible to automate the process by using AI techniques, such support is missing from current KDD tools.

To overcome these problems, this thesis introduces a new approach that takes the current ideas one step further to achieve support for the entire process. The main idea behind it is to develop an Intelligent Discovery Assistant (IDA) that can automatically generate workflows. Rather than only generating all the correct workflows, the IDA is able to rank them by different performance measures. We hypothesized that the system can produce complex workflows and recommend workflows whose performance is as good as the best overall workflow.

In support of this thesis, we developed eProPlan, a tool that is able to model, test and maintain the KDD domain for planning. The tool is based on a Data Mining ontology that represents a detailed model of the KDD domain. We then designed a novel approach on recommending KDD workflows that combines Collaborative Filtering with auto-experimentation. We explore memory-based, model-based and content-based CF approaches in the context of datasets and workflows. For new datasets or the cold-start problem, we compare several strategies from CF that allow selecting training workflows.

We evaluated our approach on classification and regression datasets. The results show that the system is able to recommend good performing workflows. Our IDA takes the capabilities of data analysis tools at a higher level providing first the possibility to automate the KDD process and second a ranker which recommends workflows that perform similarly to the best workflow. Consequently, the work presented in this thesis takes the support for KDD one step further to real support where users can

easily process their data.

Zusammenfassung

Wenn es heutzutage, im Informationszeitalter, darum geht, die ständig wachsende Datenmenge zu analysieren, spielt KDD (Knowledge Discovery for Databases) eine grosse Rolle. Man benötigt effiziente Datenanalysetools, die den Ablauf der Analyse vereinfachen und zudem gute Ergebnisse liefern. Eine der grössten Schwächen solcher Tools ist die fehlende Unterstützung für den gesamten Analysevorgang. Die grosse Menge verfügbarer einzelner Methoden als Lösungen für Teilprobleme sorgt zudem für eine Verschärfung dieses Problems. Vergangene Forschung hat bereits gezeigt, dass Techniken der künstlichen Intelligenz eine Automatisierung des Analysevorgangs ermöglichen. Bei derzeit verfügbaren KDD Tools dagegen fehlt diese Unterstützung.

Diese Arbeit stellt einen neuartigen Ansatz vor, um diese Probleme zu lösen. Dabei werden bestehende Konzepte erweitert, um die Unterstützung für den gesamten Analysevorgang zu ermöglichen. Die wesentliche Idee ist es, einen Intelligent Discovery Assistant (IDA) zu entwickeln, der automatisch korrekte Workflows generieren kann. Ausserdem ist er in der Lage, diese Workflows zu beurteilen, indem er sie nach verschiedenen leistungsbezogenen Kriterien hierarchisch ordnet. Wir nahmen an, dass das System komplexe Workflows produzieren und diejenigen vorschlagen kann, deren Leistung so gut ist, wie die des besten Workflows.

Um diese Hypothese zu stützen, entwickelten wir eProPlan, ein Tool, das in der Lage ist, die KDD-Domäne für die Planung zu modellieren, zu testen und zu pflegen. Dieses Tool basiert auf einer Data-Mining-Ontologie, die ein detailliertes Modell der KDD-Domäne darstellt. Weiterhin haben wie einen neuen Ansatz für die Empfehlung von KDD Arbeitsabläufen entwickelt. Anschliessend haben wir einen neuartigen Ansatz für die Empfehlung von KDD-Workflows entwickelt. Dieser Ansatz kombiniert Collaborative Filtering (CF) mit automatischen Experimenten auf den Workflows. Wir untersuchen speicherbasierte, modellbasierte und inhaltsbasierte CF-Ansätze im Kontext von Datensätzen und Workflows. Für die Analyse neuer Datensätze oder das Kaltstartproblem vergleichen wir verschiedene Strategien der CF, anhand

derer man die besten Trainingsworkflows auswäholen kann.

Wir haben unseren Ansatz anhand von Klassifizierungs- und Regressions-Datensätzen evaluiert. Die Ergebnisse zeigen, dass das System in der Lage ist, leistungsstarke Workflows zu empfehlen. Unser IDA bringt die Fähigkeiten von Datenanalyse-Tools auf ein höheres Niveau und liefert neben der Möglichkeit, den KDD-Prozess zu automatisieren auch ein Bewertungssystem, das Workflows vorschlägt, die ähnlich gut performen wie der beste Workflow. Es lässt sich also feststellen, dass diese Arbeit die Unterstützung für KDD effektiver macht und den Nutzern damit die Möglichkeit bietet, ihre Daten auf einfache Weise zu verarbeiten.

Acronyms

AI Artificial Intelligence

API Application Programming Interface

ARS Algorithm Recommender System

AUC Area Under Curve

CBR Case-Based Reasoning

CF Collaborative Filtering

CP Constraint Programming

CSP Constraint Satisfaction Problems

DM Data Mining

DMA Data Mining Assistant

ES Expert System

HDMA Hybrid Data Mining Assistant

HTN Hierarchical Task Network

KDD Knowledge Discovery for Large Databases

IDA Intelligent Discovery Assistant

MD Meta-Data

Meta-L Meta-learning

ML Machine Learning

MLT Machine Learning Toolbox

PDAS Planning-based Data Analysis Systems

PIP Proximity Impact Popularity

RM RapidMiner

ROC Receiver Operating Characteristic

WCE Workflow Composition Environment

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

Our society struggles with the amount of data that is produced at fast pace [Manyika et al., 2011]. Data appears everywhere under different forms (e.g., text, images, audio, video, etc.), but data is valuable only if one can make sense of it by identifying useful information to be used for interpretation and learning. Knowledge Discovery for Large Databases (KDD) addresses this issue by extracting useful information out of large datasets [Fayyad et al., 1996b] using techniques from Machine Learning (ML) or Statistics. The advances in the last years have created more and more algorithms that have been integrated in current KDD tools (RapidMiner[1], WEKA[2], Oracle Data Mining[3], IBM SPSS Modeler[4], etc.). Most of the tools offer various features that simplify the task of the user (e.g., manual design of workflows, help features, etc.). However, the user support is rather limited requiring experience and knowledge about each algorithm and also about how to combine several algorithms to build a well-performing workflow. As such current KDD tools require the users to design workflows manually.

[1] http://rapid-i.com/content/view/181/190/

[2] http://www.cs.waikato.ac.nz/ml/weka/

[3] http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html

[4] http://www-01.ibm.com/software/analytics/spss/products/modeler/

To address these issues researchers have proposed to use automatic advising systems (e.g., [Engels, 1996, Bernstein et al., 2005, Charest et al., 2008, Žáková et al., 2010, Kietz et al., 2010a]). The main idea is to have a system that handles all the stages of the KDD process – essentially an automated IDA [Bernstein et al., 2005]. This requires to identify all the rules and knowledge needed to build correct KDD workflows. However, enumerating all the correct workflows only partially satisfies the needs of a user. Too many correct workflows create a dilemma since the user does not know which workflow to choose. Therefore, a workflow ranking approach is required to recommend workflows based on several criteria (e.g., performance, execution time, etc.). The result brings us one step closer to real support for KDD since it can recommend users the TOP K workflows for a dataset based on their performance. In this way the user is guaranteed to execute a well-performing workflow without having to know everything about the way algorithms interact to compose feasible and correct workflows.

This chapter is organized as follows: In Section 1.1 we present the motivation of this dissertation by identifying the main limitations of the state-of-the-art in recommending and ranking KDD workflows. Section 1.3 introduces the overall research problems and hypotheses to be addressed in this thesis. Section 1.4 enumerates the main contributions of this work while Section 1.5 illustrates and shortly describes the structure and the main parts of this dissertation.

1.1. Motivation

The advances in technology have led to a new trend that now contours itself as the 'Age of Big data' [Lohr, 2012]. KDD is nowadays a 'must-have' and 'must-do' as studies have shown that companies which use analytics are more competitive and successful [LaValle et al., 2011]. But KDD by itself is a complex process that requires time and expertise to learn and understand all the underlying analysis techniques.

Improving the KDD workflow design experience is an active field of research. Researchers have explored different technologies to automatically generate KDD workflows [Amant and Cohen, 1995, Wirth et al., 1997a, Bernstein et al., 2005, Choinski and Chudziak, 2009, Žáková et al., 2010]. Data Mining (DM) ontologies have been developed, but most of them only cover some parts of the KDD process (only some of the steps or only few algorithms) [Diamantini et al., 2009a, Charest et al., 2008, Žáková et al., 2010] or are very generic and do not focus on the KDD workflow [Panov et al., 2008]. Even more some of them were only used as a proof of concept

and are not available anymore [Bernstein et al., 2005]. To be able to generate only correct KDD workflows one needs to model data characteristics at a fine level of detail. Most of the existing ontologies can only generate very simple KDD workflows that consider only basic characteristics of the data [Bernstein et al., 2005, Diamantini et al., 2009a, Žáková et al., 2010]. To automate the KDD process one needs a detailed description of the steps and algorithms. Therefore, there is a dire need for developing a complex ontology for DM with focus on the KDD process. Moreover, as KDD is a dynamic field and new algorithms appear every year the ontology needs to be improved and extended over time. Such tools have not been developed or designed before. This is the first limitation of state-of-the-art research.

Recommending the right method for data analysis was explored extensively. Researchers have tried to identify situations and characteristics of datasets that allow them to define rules for choosing the right technique. However, the formulation of such rules is very difficult or even impossible [Schaffer, 1994, Wolpert, 2001]. The existing research mainly focuses on the DM step and not on the whole KDD workflow. The idea is to find the best performing algorithm on a given dataset based on the dataset's characteristics. People have not explored the entire workflow and therefore omitted other steps like preprocessing, transformation, feature selection, etc. Thus, there should be a switch from algorithm to workflow that focuses on all the steps of the KDD process. We consider this an important future change in the KDD domain moving the focus from the current extensive development of better performing methods to a thorough exploration of how these methods should be combined to give a better outcome of the analysis. We identified this as the second limitation of the state-of-the-art research.

The existing approaches for automation of KDD workflows only generate a relatively small number of workflows. Additionally, most often no ranking is provided (with the exception of [Bernstein et al., 2005, Nguyen et al., 2012]) requiring to execute all the workflows to figure out which one works better. Hence, most often there is no selection or choice strategy available. This is another limitation of the current research.

In this dissertation we address the major limitations identified from the state-of-the-art research. First, we identify the main problems of the existing IDAs. We explain the main concepts of the workflow ontology used for automating the KDD process. We introduce the tool responsible for building and modelling the ontology as well fas for retrieving all the possible correct workflows. Second, we provide a framework for ranking the workflows and recommending the Top K workflows. Finally, we analyze the experiment case-base and identify workflows and datasets for which pre-processing

improves or decreases the performance of a workflow. The pre-processing study compares workflows without certain pre-processing steps to the ones with the respective pre-processing steps to assess the impact of certain pre-processing steps on the final result.

In the remaining part of this chapter we first introduce the terminology that helps us define the main terms required for a good flow and understanding of this thesis. Then, in the next part we introduce a set of research questions and hypotheses that are tackled in this work.

1.2. Terminology

For a clear understanding of the problems that we are solving first we introduce the concepts of *operator*, *workflow*, *correct* or *valid workflow* and define an *optimization objective*.

Definition 1 *An operator is an ML algorithm used for analyzing or processing data.*

Operators have inputs–one or more datasets, outputs and parameters used to tune their behavior (most often depending on the implementation). They can also be grouped in hierarchies based on their functionality. This allows to have abstract operators–with no actual implementation and basic operators–with real implementations and which can be applied on the data.

Definition 2 *Given a dataset d and a DM task t, a workflow is a sequence of operators $o_1, o_2, ..., o_n$ that tries to transform the initial dataset d in order to solve the task t.*

Workflows or processes are a sequence of operators where for each step of the KDD process there is a corresponding operator.

Definition 3 *A correct or valid workflow is a workflow that contains only matching operators in the sense that the output of the operator o_i matches the input of the operator o_{i+1}.*

Valid workflows most often do not generate exceptions during execution [5]. However, one can make a verification by adding semantics to each of the operators.

As our main objective is to rank workflows in terms of their performance on datasets we define the optimization objective used for this process.

[5] For some operators is not possible to detect all problems beforehand.

Definition 4 *The optimization objective of the ranking process is to maximize the performance of workflows in terms of accuracy for classification problems and minimize the regression error for regression problems.*

Other functions can be used as for example the running time or a mixed function of time and performance.

1.3. Research Questions and Hypotheses

The overall goal of this thesis was to develop a system that is able to generate, rank and recommend KDD workflows to users [6]. To achieve this we have built/extended a KDD workflow ontology and a system that allows to automatically generate all the correct workflows. As a next step we focused on how to rank the workflows by combining different strategies and methods from collaborative filtering with auto-experimentation. This represents the main part of the thesis.

Currently more and more KDD tools are available that incorporate more and more algorithms. They allow users to analyse data by building KDD workflows using their GUIs (Graphical User Interface). However, the amount of algorithms is overwhelming. To build correct workflows users need knowledge about the strengths and weaknesses of each algorithm as well as their influence on the next steps. For this reason, Research Question 1 (RQ1) tries to identify the problems of existing IDAs.

RQ1 *Why is KDD difficult for users and what are the main issues of today's tools?*

RQ1 addresses the main difficulties a user experiences when trying to build KDD workflows. We analyse and discuss the following hypothesis related to RQ1:

HYPOTHESIS 1.1 *Current KDD tools have a limited support for users.*

Existing KDD ontologies are either a proof of concept or were only used to identify important concepts from the KDD domain. We take the idea from [Bernstein et al., 2005] further and identify the main requirements for planning KDD workflows. We propose a mapping between the planning components (as defined in classical planning) and the KDD domain. Here, the planning problem, initial state, goal and planning actions are described in detail.

[6] This thesis is not based on user studies. We rely our thesis only on findings from a literature survey and empirical results.

RQ2 *What characteristics should a KDD workflow ontology have to allow automatic generation of workflows?* [7]

RQ2 explores what characteristics/structure an ontology should have such that it can be used for planning KDD workflows.

HYPOTHESIS 2.1 *Ontologies and planning can be used the generate automatically all the correct workflows.*

The KDD domain is very dynamic and changes at fast pace since new algorithms and types of data are appearing. As such, a KDD ontology cannot ever be complete and needs to be updated with new algorithms. As the ontology describes the domain for planning normal ontology editors are not sufficient since they do not focus on the planning components but only on generic ontology features. This raises the question of how to easily edit, test and maintain an ontology used for planning.

RQ3 *How could the ontology be maintained and modeled over time?*

We analyze and discuss the following hypothesis related to Research Question 3:

HYPOTHESIS 3.1 *The ontology needs to be updatable such that new algorithms can be added.*

Being able to generate all correct workflows for a given dataset does not satisfy completely the user support. This only produces a multitude of alternatives and avoids the manual building and interleaving of operators for each step of the KDD process. We assume that users in general want a limited number of recommendations as for search engines or products in general. We assume that they can restrict their workflow search and ask for best workflows that fulfill a certain criteria (e.g.., best 1, best 3, best 10). How can this be achieved to get a good ranking such that the recommended bet workflows are close to the real best ones? Research Question 4 analyzes this issue and explores several possible solutions.

RQ4 *How could one rank KDD workflows and recommend Top K workflows?*

We analyze and discuss the following hypothesis related to Research Question 4:

[7] Was joint work with several people in the e-Lico project (Jörg-Uwe Kietz, Simon Fischer). We include it for the flow and to understand how the eProPlan system works.

HYPOTHESIS 4.1 *Collaborative filtering and Auto-experimentation can be used to rank the generated workflows.*

Important parts of a workflow are the pre-processing steps, filling missing values, type conversion, and normalization. Many researchers have argued that pre-processing is crucial for getting the best results out of the data at hand. Based on the auto-experimentation results we analyze this research question and compare different workflows based on their structure and the datasets they were applied on.

RQ5 *How important is pre-processing in the KDD workflow?*

HYPOTHESIS 5.1 *Pre-processing is improving the performance of workflows.*

1.4. Contributions

By analyzing the research questions we made several contributions affecting the Data Mining and Data Analysis fields. Our contributions can be summarized as follows:

- We have realized an extended survey of the state of the art tools that improve the user support during the data analysis process (IDAs). This summarizes 30 years of research in the field of intelligent assistance for data analysis. It provides an overview of the field as well as a classification and comparison of IDAs in different categories based on their capabilities.

- Having identified the problems of IDAs we developed eProPlan, a system that uses planning and ontologies to automatically generate all correct KDD workflows. The system also supports the development and maintenance of the ontology over time. New operators can be easily modeled and checked for correctness. Useless workflows are avoided by employing Hierarchical Task Network (HTN) planning which allows to define the main steps of workflows. We have also developed an IDA-API that enables to plug-in the capabilities of the planner into any KDD tool. We introduce the DM ontology as its at the core of the system and represents the planning domain. However, the ontology is not a contribution of this thesis but it is the work of several people and was slightly modified for some of the experiments.

- Only recommending all the possible workflows for a given problem is not enough. Most often the number of generated workflows is very large. Therefore, we have developed a ranking system to rate the workflows by their performance using

collaborative filtering techniques and auto-experimentation. Based on this approach we can recommend users the Top K workflows and ensure that at least one workflow is comparable in performance to the best existing workflow.

- Auto-experimentation provides us the performance details of several workflows resulted by valid/correct combinations of algorithms. One of the main reasons of using workflows and not algorithms was supported by research in the field that arguments the importance of pre-processing and attribute selection. We have designed a study to compare the performance of workflows in the presence and absence of specific pre-processing operators. This allowed us to understand under which circumstances some workflows work better or worse than without pre-processing.

1.5. Thesis Outline and Foundation

This thesis is structured in several parts as follows:

Next chapter presents a survey of intelligent data analysis systems over the last 30 years of research [Serban et al., 2012]. Several approaches have tried to improve and simplify the data analysis process for users. Based on an extensive paper survey we identify the types of support developed by scientists over the past years: support for single steps, support for multiple steps, graphical design, automatic workflow generation, etc. We categorize the type of knowledge needed to achieve the support: this involves the characteristics of the data, the algorithms used, the models produced, and the history of already run experiments. Then we group the IDAs in categories based on their main feature, e.g. expert systems, meta-learning systems, case-based reasoning, planning and workflow environments. The characteristics of each of these systems are presented. Based on this classification we compare and discuss the type of support and knowledge to identify the systems' usefulness. Finally, we discuss the characteristics of an ideal IDA that should incorporate all important features identified in this survey.

A skeleton of the thesis with the main chapters and research questions can be seen in Figure 1.1.

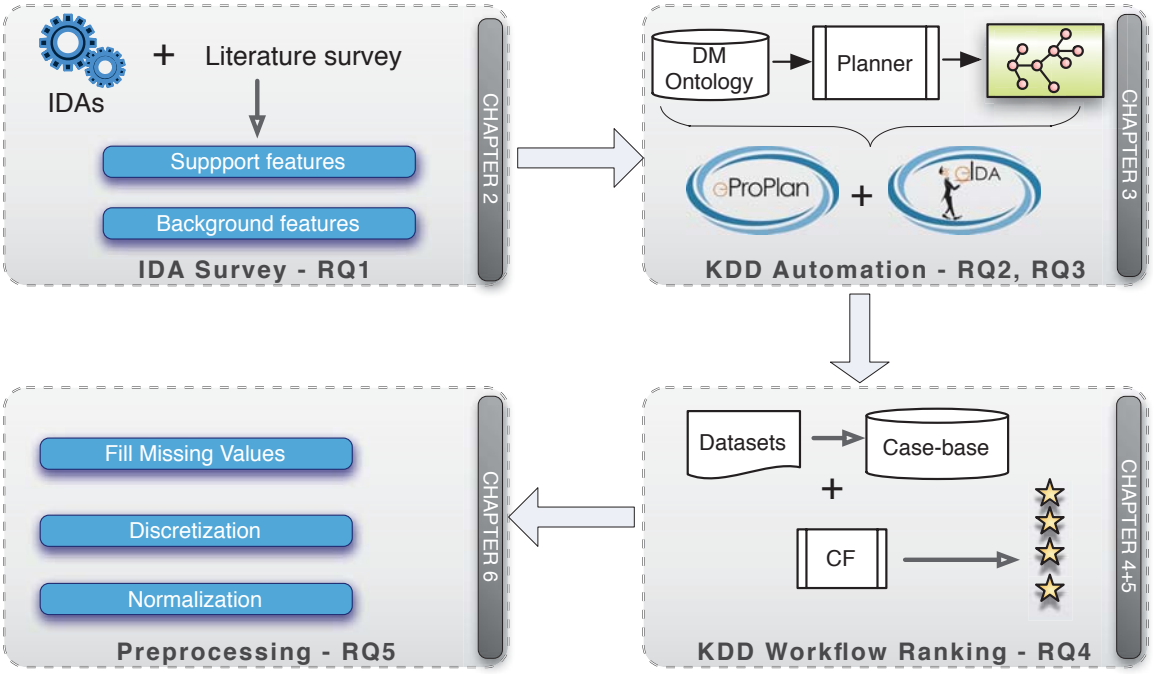


Figure 1.1.: Overview of the thesis.

This chapter addresses the first research question that tries to identify the main problems of existing IDAs. It handles and supports the first hypothesis.

Chapter 3 represents an extension of the [Kietz et al., 2012, Serban, 2010] and it is also based on [Kietz et al., 2009, Kietz et al., 2010b, Kietz et al., 2010a]. It describes how eProPlan works and what are its main components. The system relies on the KDD workflow ontology [Kietz et al., 2012] that defines the main components of a KDD workflow and its main steps. To understand how the workflows are generated we describe the HTN-planner and the planning algorithm. The system's main features are presented: editor for operators, editor for tasks/methods, editor for new built-ins, applicable operators, etc. Moreover, to profit even more from eProPlan we have developed an IDA-API that allows to plug-in the feature of automatically generating workflows in any KDD tools (e.g., RapidMiner and Taverna plug-ins already exist).

This chapter handles RQ2 by describing the main concepts of the ontology and RQ3 by presenting the editor that allows editing and maintaining the ontology over time.

Chapter 4 introduces a set of techniques used for ranking the workflows generated by the IDA-API by combining auto-experimentation and Collaborative Filtering (CF).

First, we selected a set of datasets for which we executed all workflows and stored their execution details. Then we use various collaborative-filtering methods to rank the workflows for a new given problem by only running a small number of workflows. We explore three different methods: pure CF, content-based CF and finally a hybrid approach combining both of these. The main challenges of these approaches are: selecting the number of workflows needed for training, selecting the features of datasets and workflows used for the content-based approach, and finding a way to combine the two methods.

In this chapter we address the research question RQ3 by explaining the methods and algorithms used for ranking the plans.

Chapter 5 empirically evaluates the methods presented in Chapter 4 and discusses the results. We first introduce the experimental setup and some evaluation methods for assessing the performance of the CF methods and also the quality of the recommended workflows. This chapter provides results and support for each of the hypotheses made for research question RQ4. The results show that auto-experimentation and CF methods can be used to rank KDD workflows.

Chapter 6 explores several pre-processing methods, identifies and discusses their influence on the performance of workflows. This is based on the results from the auto-experimentation case-base. Having a large number of workflows with various combinations of operators allows us to understand the effects of pre-processing on several datasets.

Chapter 7 discusses the limitations for each of the explored research questions. Then presents a set of possible avenues for future work.

Chapter 8 concludes the overall thesis. Having explained in detail the content of the current thesis, we now take a look at the overall picture and present the impact and the main findings of the thesis. Limitations and future work are discussed to position the thesis in the current context and understand its possible extensions.

1.6. List of Papers

The thesis is based on the following papers:

- A Survey of Intelligent Discovery Assistants for Data Analysis, ACM computing Surveys, Accepted in 2012, To be Published in 2013.
  This paper was joint work with Joaquin Vanschoren, Jörg-Uwe Kietz and Abraham Bernstein. My main contribution to this paper was identifying different support criteria for IDAs and compare the systems along this line. An initial paper

with preliminary findings was published already in 2010: F Serban, J U Kietz, A Bernstein, An overview of intelligent data assistants for data analysis, In: 3rd Planning to Learn Workshop (WS9) at ECAI'10, 2010-08-16.

- KDD ontology, eProplan and eIDA
  The system is a result of the work done in the e-Lico project. This was joint work with Jörg-Uwe Kietz, Simon Fischer and Abraham Bernstein. In this project my contributions were mainly the compilation of the ontology for planning, the planning and goal editors, the IDA-API, the installer for RM-IDA, testing and debugging some of the other tools. To automatically generate the workflows we have used the planner developed in the e-Lico project. The following papers were published along this topic:

  1. J-U Kietz, F Serban, A Bernstein, S Fischer, Towards cooperative planning of data mining workflows, In: Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-09), 2009-09

  2. J-U Kietz, F Serban, A Bernstein, eProPlan: a tool to model automatic generation of data mining workflows, In: 3rd Planning to Learn Workshop (WS9) at ECAI'10, 2010-08-16.

  3. J-U Kietz, F Serban, A Bernstein, S Fischer, Data mining workflow templates for intelligent discovery assistance in RapidMiner, In: Proc of the RCOMM'10, 2010-09-13.

  4. J-U Kietz, F Serban, A Bernstein, S Fischer, Data mining workflow templates for intelligent discovery assistance and auto-experimentation, In: Proc of the ECML/PKDD'10 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD'10).

  5. J-U Kietz, F Serban, A Bernstein, S Fischer, Designing KDD-Workflows via HTN-Planning, In: European Conference on Artificial Intelligence, Systems Demos, IOS Press. 2012-08-27.
  6. J-U Kietz, F Serban, A Bernstein, S Fischer, Designing KDD-Workflows via HTN-Planning for Intelligent Discovery Assistance, In: Planning to Learn

2012, Workshop at ECAI 2012, CEUR Workshop Proceedings, 2012-08-28.

7. J-U Kietz, F Serban, S Fischer, A Bernstein, Semantics Inside! But let's not tell the Data Miners: Intelligent Support for Data Mining, In: European Semantic Web Conference ESWC 2014, Springer, 2014.

- Ranking KDD workflows
  I have developed the experimentation module and run all the possible workflows for several datasets from online repositories (UCI, DELVE, etc.). Next, I have implemented a set of collaborative-filtering techniques for ranking the workflows for new datasets. A paper that explains the auto-experimentation approach was presented at the doctoral symposium in 2011 and one is under review:

  1 F Serban, Auto-experimentation of KDD workflows based on ontological planning, In: The 9th International Semantic Web Conference (ISWC 2010), Doctoral Consortium, 2010-11-07.

2

Background and Related Work

The major contribution of this thesis is effective support for the KDD process using techniques from several fields: ontologies, planning and Collaborative Filtering (CF). We were motivated by the increasing number of algorithms in the fields of ML and KDD.

This chapter introduces the domain and presents the related work from the research field. We start by explaining the basic building blocks of this thesis. In Section 2.1 we shortly describe the KDD domain with its characteristics and goals and focus more on the KDD process and its steps. We identify a set of challenges that KDD is facing and which influence the work in this thesis. Then, we define a set of support features that could improve the KDD experience for users. In the light of these features we survey existing systems that improve and sustain the whole KDD-process. The systems are organized based on their capabilities and then compared along the identified features.

In Section 2.3 we explore in depth approaches for automating the KDD-process since we have built our work in similar directions by filling in some of the gaps. Ranking of algorithms was already proposed in Meta-learning (Meta-L). We explore the research and introduce the basic techniques.

2.1. Knowledge Discovery for Databases

Every second more and more data is produced. A key factor for the fast growth is the development of the World Wide Web that allows to store, transfer, share and access information all over the world. The recent advances in technology (memory, storage, speed) facilitate and support this process. Online stores (Amazon) and social networks (Facebook, LinkedIn, Google Plus) have billions of users which generate enormous amounts of data every day. Other important areas that produce data are banking, telecommunications, and the retail industry. The raw data is stored in different formats (structured large databases or other formats like images, plain text files or semi structured XML files, etc.). The main challenge is how to extract relevant information out of it. This is the task of Knowledge Discovery for Large Databases (KDD) [Frawley et al., 1992, Fayyad et al., 1996b]. The concept of KDD was proposed and introduced to solve exactly this problem. As more real-world applications are striving through huge amounts of data, KDD becomes a key factor in industry as well as in research [Fayyad et al., 1996a].

KDD is an umbrella-like term that relies on techniques from several areas of expertise: ML, statistics, pattern recognition, databases, Artificial Intelligence (AI), knowledge acquisition for expert systems, data visualization and high-performance computing. It is a very dynamic and interdisciplinary field that evolves with the nature of data – new types of data mean new methods to process it. These fields provide some of the techniques that are used either in the DM step or in other steps (interpretation and visualization). The difference between KDD and all the underlying areas of expertise is the fact that it focuses on the overall process of knowledge discovery from data including how it is stored, accessed, processed, interpreted, visualized and explained to the end user.

2.1.1. The KDD-process

KDD is the process of analyzing data by extracting patterns and valuable information out of it. We adopt the academic KDD definition of [Fayyad et al., 1996a]: “KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data”. It represents a template for any kind of data exploration that applies a set of transformation steps from raw data to the actual useful knowledge. There are several perspectives on KDD some with more focus on the application of KDD to real domains (CRISP-DM, SEMMA, etc.) [Brachman and Anand, 1996, Azevedo, 2008, Cios et al., 2010]. However, the most common is the one from

[Fayyad et al., 1996a] and we will rely on it for the purpose of this thesis. As shown in

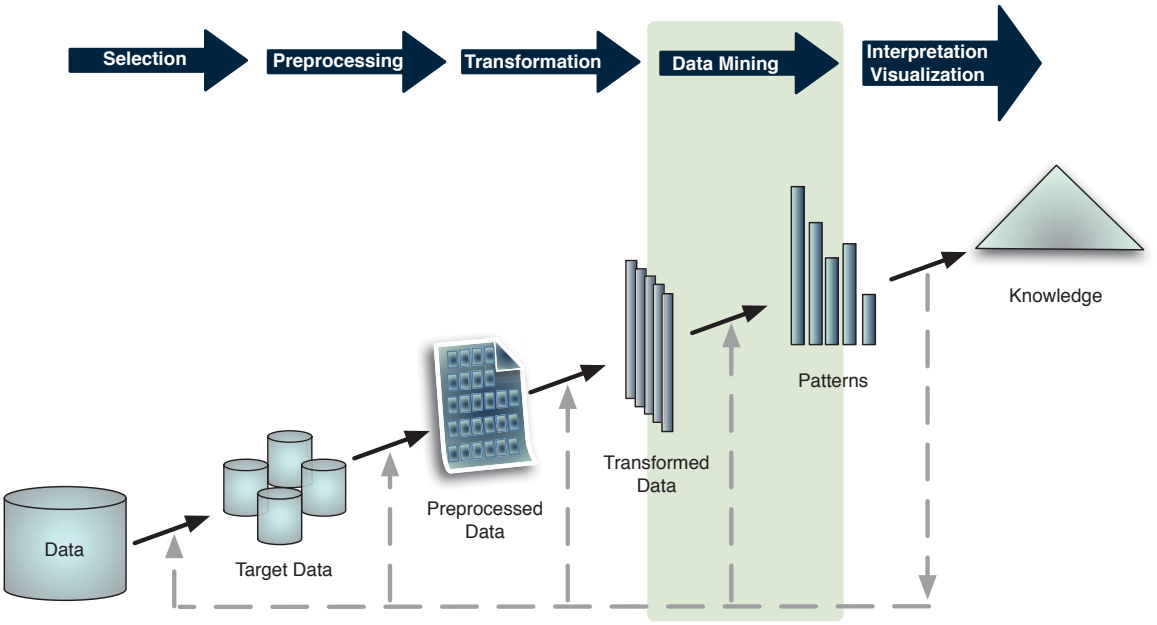


Figure 2.1.: Overview of steps from the KDD-process. Adapted from [Fayyad et al., 1996a]

Figure 2.1, the process consists of the following steps:

DATA SELECTION The first step selects from the raw data the focus of the knowledge discovery process: the target data. Most often raw data is stored in databases but nowadays this has been extended to other sources of data (images, text, etc.)[1]. This consists of a set of samples and variables that are used for the analysis.

DATA PREPROCESSING Cleaning and preprocessing is an important step. Here noise is removed and different strategies are applied to fill missing values. Not handling this step correctly can seriously damage the final outcome of the KDD process.

DATA TRANSFORMATION The dimensionality of the preprocessed data is reduced: correlations between attributes are found and only most important ones are kept. Feature selection and data sampling are common transformation techniques.

[1] This new types of data can be transformed into relational data by extracting relevant features as columns.

DATA MINING This is the core of the KDD-process and consists of three steps: first the goal of the KDD process is matched to a DM or statistical modeling technique. Depending on the main goal, different supervise or unsupervised methods can be selected (classification, regression, clustering, association rules, etc.). Second, the right DM algorithm and its parameters are selected. Third, the algorithm is applied on the data and different forms of patterns are generated.

EVALUATION The last step is also more complex. First, patterns, models and data are visualized and interpreted. Finally, in a follow up step the incorporated knowledge may be used for taking decisions, documentation or reporting or it may be incorporated into other systems. This is the exploitation step which allows to use the learnt model to make predictions on new data.

The KDD process includes several iterations and it may even involve loops between some of the steps (e.g., cross-validation is a loop over feature selection and data mining).

2.1.2. Challenges

KDD was born from the need of organizing and formalizing the data analysis process. Its main purpose was to provide a set of guidelines on how to analyze large amounts of data. Even if KDD has evolved over time, many issues are still open. Next sections present shortly two of the main confrontations of KDD nowadays: the size of the data and the large number of techniques.

2.1.2.1. Growth of data

20 years ago data was growing already at a fast rhythm: “It has been estimated that the amount of information in the world doubles every 20 months” [Fayyad et al., 1996b]. What about today? A study from Thompson Reuters presumes that by 2015 data will explode [Reuters, 2012]. Now we have storage capabilities to save a lot of data. The main question is how to analyze this data effectively. This is an open question that many people both from research and industry are trying to solve [Manyika et al., 2011]. New techniques suitable for large data are being used (e.g., Map/Reduce paradigm [Dean and Ghemawat, 2008] based on its implementation Hadoop [2] for RapidMiner, the so called Radoop [Prekopcsák et al., 2011]).

[2] http://hadoop.apache.org/

2.1.2.2. Growth of algorithms

As new types of data have appeared new algorithms have been developed to analyze them. Therefore, the number of algorithms has grown for each step of the KDD process and thus the number of possible workflows easily explodes. For example, if we look at RapidMiner with all its plug-ins it reached more than 1000 operators. Even more, the results from the user survey of RapidMiner operator usage show that a large variety of algorithms is used as shown in Fig. 2.2. Most of the operators have a similar usage frequency and only a few operators are highly used (e.g., DecisionTree, FillMissingValues, SVM, NaiveBayes, etc.) [3]. This shows that a serious problem of KDD is the multitude of options that users have without having the capabilities to reason about which one is the best choice for a given task at hand.

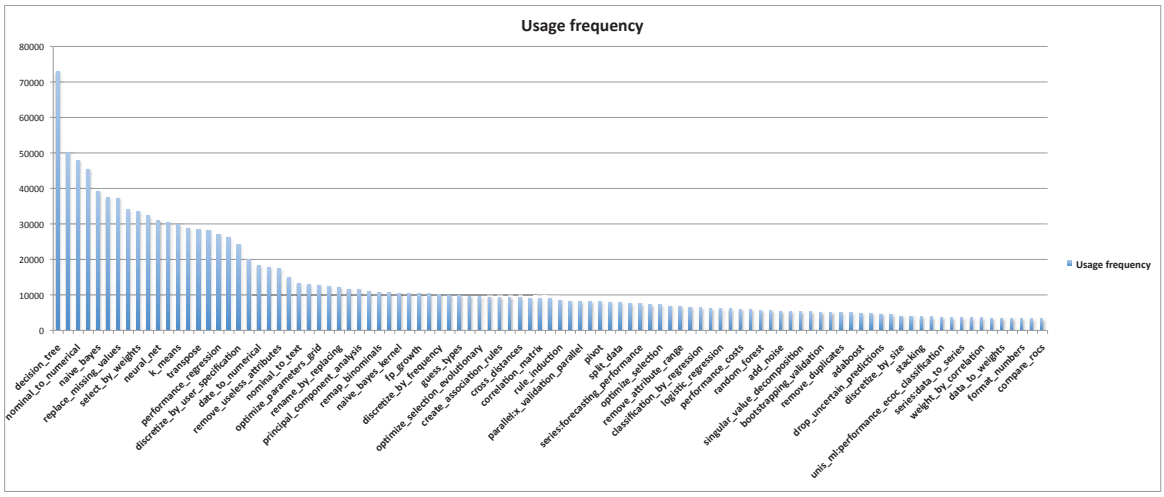


Figure 2.2.: RM Operator usage frequency

In this thesis we explore in depth the main questions raised by the second issue, the large number of operators. The first issue is also very interesting, but it is not part of the scope of this thesis. In the following section, we analyze the main tools that support users during the KDD process and identify a set of features that make such tools assistants for the data analysis process.

[3] We excluded IO operators (reading, writing data) and operators like Multiply, Join, etc.

2.2. Towards User Support for KDD

[4] KDD is an interactive process that requires constant feedback from the user [Brachman and Anand, 1996]. Before designing a KDD process users need to understand their data and the purpose of the analysis in depth. This is the key to a successful analysis. Data analysis is quite an old field and during time many tools have evolved and matured. We have performed a survey over the last 30 years of research about IDAs to identify what kind of support do the tools use to guide the users during the KDD process [Serban et al., 2012]. In this section, we present some of these findings with special focus on support and comparison of the systems.

2.2.1. Types of support

Support for users analyzing the data could be incorporated in each of the KDD phases, in a combination thereof, or even in its entirety. It can help users with the construction of an analysis, with its execution, or with its understanding. In this section, we explore all these types of support developed by scientists over the past, in order to gather the set of requirements for an IDA.

Support for a single step of the KDD-process Some systems help the users to model a single step of the KDD-process by guiding them through the process of applying a specific operator. They provide hints and advice on selecting the right type of input data (e.g., nominal, scalar, ordinal) including parameter selection and tuning. Several algorithms can be used for each step of the KDD-process. The challenge, however, is to find the right one for the given data and analysis task at hand. Novices are usually overwhelmed by the plethora of approaches and tend to revert to only a few known techniques. But even experts are often unaware of suitable, less-known methods as shown in [Kohavi et al., 2000, Bernstein et al., 2005]. Moreover, each algorithm owns several parameters; therefore, it acts differently if diverse parameter values are chosen. Thus, one needs to find the appropriate values for the parameters to get the optimal output on a certain data set, since the data set influences the process of choosing the best matching parameters. Lastly, some techniques have specific requirements on the form of data; as for example the ID3 tree induction algorithm uses only nominal attributes. Therefore, stepping back to the data preparation phase is often needed. Hence, support in choosing the adequate algorithm and pa-

[4] Parts of this section are from our ACM Computing Survey paper [Serban et al., 2012]

rameters for a given analysis need is clearly desired.

Support for multiple steps of the KDD-process A less common feature in IDA systems is the support for the overall KDD-process. Here the system provides useful help regarding the sequence of operators in the KDD-process as well as their parameters. Hence, it helps in addressing the overall data analysis process from raw data to the actionable knowledge. In order to be able to support the whole process, the system must have knowledge about the usage of each step in correlation to the other ones. More precisely, it has to know what operators can be applied in which situations as well as how to set the parameters of those operators. This knowledge is extracted from DM cookbooks, such as the CRISP-DM [Chapman et al., 1999] or from experienced users. Note that effective support for the overall process should go beyond the mere construction of correct processes. It may need to consider domain- and task-specific knowledge as well – a capability that has eluded AI-based approaches and required human creativity in analyzing data so far.

The support for combining multiple steps (e.g., auto wiring, correction proposals, next operator recommendation) does not imply that they are planned fully automatically.

Graphical design of KDD workflows Graphical editing refers to enabling the user to interactively build the process manually: choosing the operators and setting the right inputs/outputs and parameters. Some of these GUIs integrate intelligent techniques, such as auto-wiring, meta-data propagation, correctness checking before execution, and operator recommendation. Even without such techniques, they can be viewed as a baseline for DM as it is used today and which has to be improved with intelligent assistance. They are often seen as being user-friendlier than textual representations of the KDD-process. Graphical editing systems allow the users to drag and drop operators and connect them via inputs/outputs (called ports). Such systems use either implicit data objects (no representation) or explicit ones. Often, several tabs are used for displaying information to the users as well as pop-ups, graphs, etc. In addition, different layouts and colors are employed for displaying the data as well as meta-data and its propagation.

Automatic KDD workflow generation A special case of user support for multiple steps of the KDD-process is the automatic generation of workflows. Here the system provides the users with one or more KDD-processes that they need to execute in order to solve the DM task. The system automatically sets all inputs/outputs and

parameters for each operator. This is especially useful for users who do not have a lot of experience with DM operators. Based on the data and a description of their task, the users receive a set of possible scenarios for solving a problem.

Automatic generation is a much more challenging problem than the previous ones since it requires knowledge about the order of the steps in the KDD-process. Usually the knowledge is extracted from DM cook-books and from expert data miners. In general, automatic generation involves a number of issues that need to be addressed:

- *How compatible is the generated workflow with user-modifications?*
  It would be useful if the user were able to make modifications to the generated workflows and execute them. The user feedback should be maintained.

- *How good are the generated KDD workflows?*
  The quality of a workflow is defined both by its correctness in execution and its performance with respect to evaluation criteria such as accuracy and speed. In addition, the systems can either "just" list the best performing workflow or rank all of the constructed ones according to various evaluation criteria.

- *How many workflows are generated?*
  Some systems generate only one solution (as in classical planning); others try to generate all possible solutions.

- *Which ones are generated and why?*
  Does workflow generation follow certain rules or heuristics? How are these chosen?

Many practitioners and experts believe that the KDD-process cannot be fully automated since it is a creative process and the user has to constantly check the status of the workflow. At some stages of the workflow, only a human user can decide what is correct or not. Today's systems, for example, encounter major difficulties in identifying the correct data-type for some attributes (e.g., a number should be interpreted as either nominal or ordinal or scalar?).

Workflow checking/repair Automatic workflow checking is not only an important feature of data analysis systems but also a recommended feature for IDAs. It checks a user-designed workflow for correctness and displays potential problems. For example, the system checks whether the port type is correct, whether the input data has the right type, etc. Additionally, another useful feature is the system's ability to repair

a workflow or suggest fixes for generating correct workflows. The system can automatically connect the right ports, suggest what is wrong and what the user should do in order to solve the problems.

Task decomposition As shown in knowledge acquisition, task decomposition structures simplify large processes [Chandrasekaran et al., 1992]. Some IDAs, hence, allow modeling KDD-processes using task decomposition primitives [Engels, 1996]. Task descriptions can be reused, thus decreasing the development time and simplifying the process of decomposing a KDD task.

User design support Additionally to providing a graphical editor, IDAs can display information about operators and their usage akin to tool-tips in modern Integrated Development Environments such as Eclipse. They allow users to easily obtain information about operators for solving problems or errors. Furthermore, some information can pertain to multiple steps: CRISP-DM [Chapman et al., 1999], for example, provides guidelines for structuring a KDD-process by reminding of what to do when and how.

Explanations regarding a decision or a result Whilst IDAs may offer assistance in the KDD-process, it is important that they provide the rationale for the assistance. The rationale is the basic building block for aiding the users in sense-making [Russell et al., 1993] and allowing them to reason about the aid provided by reflecting on whether they wish to accept the help or proceed differently. Additionally, some IDAs also support the interpretation of results by automatically providing graphs thereof or choosing other appropriate representations.

Execution of operators/workflows Some IDAs allow the users to execute workflows they created or which are generated automatically by the system. Not all IDAs implement their own operators; some only make use of existing statistical or ML libraries. Other IDAs inform the user about the usage of resources and provide estimations about the time needed to execute operators on data sets as well as the space used (i.e., the main storage for the processing and the disk-storage for the results).

For the systems that support automated generation of workflows, execution also includes experimentation in the design process for finding what kind of workflows perform better for which data sets. The experimentation results are further used to provide workflow rankings by different criteria (i.e., accuracy, execution time, etc.).

DM experience level Another attribute of IDAs that has been considered by Hand is the user's prior experience [Hand, 1985]: *Who is actually using the IDA?* The users can be either DM experts or naïve users (experts in other fields like biology, genetics, etc.) but have less knowledge about DM. Depending on the type of user considered, the system needs to provide different kinds of support.

Naïve users typically encounter two kinds of problems: First, since their understanding of DM concepts is limited, it is difficult to decide which DM operators to choose when analyzing their data. Second, it is difficult to determine which sequence of operators to apply as well as how to interpret the results obtained. Hence, users need to be led through the sequence of steps they have to apply, subsequently they need to be explained how the decisions were made and why a certain operator was selected. Above all, the system should protect the users from misuse, mistakes and misunderstanding of the DM terms.

Experts, on the other hand, have different needs and requirements. They already know how to use the main DM operators, how to modify their parameters, and how to gain a better performance. They rather need help in discovering new and more appropriate operators and in overcoming inertia at choosing the adequate approaches. In addition, the combination of operators with their parameters sometimes results in too many possible solutions, and therefore, it is impossible for a human to find the appropriate sequence. Hence, an IDA could help by suggesting workflows that were ranked according to different criteria. The experts can then compare their own solutions and validate their own decisions. Usually, experts do not want to be slowed down by being asked too many elementary questions or be led inflexibly through steps. They prefer to be able to skip support at any time or even modify a certain step. The ideal IDA should leave the expert to direct the decisions and provide explanations only on demand.

Other types of support We introduce a set of features that are not essential for an IDA but which are recommended to have since they improve the users' experience. *Visual exploration* is a necessity for a system that performs data analysis. Most of the existing KDD tools as well as exploratory IDAs (AIDE) offer different visualization facilities like icons, interactive graphs, interactive tables, etc. As suggested in [Theus, 1998], *interactivity* is the key to a good explorative data analysis tool, thus it is a desired feature for an IDA as well.

Mixed-initiative interaction As described in [Brachman and Anand, 1996] KDD is a very interactive process that needs constant feedback from the user. The IDA

should combine the techniques from both mixed-initiative user interfaces [Horvitz, 1999, Hearst, 1999] and mixed-initiative planning. As argued in [Hearst, 1999] the mixed initiative interaction is an essential feature of systems that perform complex tasks as our IDA is doing. Moreover integrating an automated system together with direct manipulation (user's actions) can generate a more natural collaboration between humans and computers.

Improved navigation plays an important role in user performance. Special tabs, editors should be designed such to decrease the time needed to solve a certain KDD task. Grouping of actions based on their scope or users' knowledge may help.

User preferences can be incorporated to improve the user experience. For each user, logs may be kept with the history of previously used workflows, operators or performed actions. Recommendations can be made based on this data.

Each type of support relies on different types of knowledge: *operators* – the algorithms that are used to analyze the data, *characteristics of the data* - the type of attributes, missing values, noise, *characteristics of operators* – the inputs and outputs, parameters and conditions and effects describing when an operator can be applied and what does it produce. It can be useful to store a set of template workflows or successful workflows in a *case-base*. All the types of knowledge are described in detail in [Serban et al., 2012].

2.2.2. Classification of KDD systems

This section provides an overview of the various IDAs developed to date. We focus on the systems that provide the user with intelligent guidance for designing workflows and will skip systems that merely offer tools but no specific guidance. We categorize these IDAs into five categories based on the core technique used to generate useful advice:

- *Expert systems* apply rules defined by human experts to suggest useful techniques
- *Meta-learning systems* automatically learn such rules from prior data analysis runs
- *Case-based reasoning systems* find and adapt workflows that were successful in similar cases

- *Planning systems* use AI planners to generate and rank valid data analysis workflows
- *Workflow composition environments* facilitate manual workflow creation and testing

2.2.2.1. Expert systems

The earliest systems designed for assisting users in data analysis were Expert System (ES)s. A high-level overview of their components is shown in Figure 2.3. Domain experts provided the domain rules that were stored in a knowledge base by knowledge engineers. These rules specify which techniques should be used and in which context. The brain of the system is the inference engine. A questioning answering module allows users to interact with the system and ask questions about a given problem. The queries are passed to the inference engine that deduces and recommends a set of possible steps/techniques based on the knowledge.

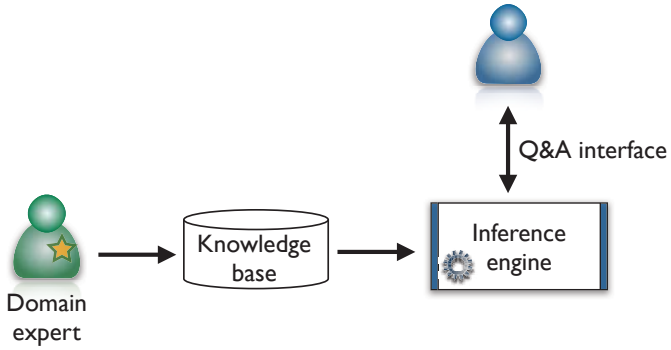


Figure 2.3.: General architecture of expert systems

Forerunners: Statistical Expert Systems The earliest systems that provided advice on data analysis were statistical expert systems, which were based on rules defining when certain statistical techniques were useful or valid. REX [Gale, 1986] covered linear regression, SPRINGEX [Raes, 1992] handled bivariate, multivariate and non-parametric statistics and Statistical Navigator [Raes, 1992] covered techniques such as multivariate causal analysis and classification. These systems asked the user multiple questions in order to trigger the expert rules. For instance, in REX the user was asked to choose between a fast result and a more accurate result based on a larger set of tests. Some other systems allowed the user to explore the rules directly to learn more about the techniques, e.g. using a textual search function (KENS [Hand, 1987, Hand, 1990]) or a help function with hyperlinks (NONPAREIL [Hand,

1990] and LMG [Hand, 1990]). In all these systems, advice was limited to the hard-coded expert knowledge and a reduced set of problems. They did not provide direct guidance, but informed the users what possible (correct) decisions could be made at the next step of the analysis. Regarding their performance, we are not aware of any study that measures how good these systems were.

MLT Consultant The first IDA for ML algorithms was Consultant-2 [Craw et al., 1992, Sleeman et al., 1995], which provides advice about the algorithms in the Machine Learning Toolbox (MLT) [Graner et al., 1993, Kodratoff et al., 1992]. The advice relied on a knowledge base with about 250 heuristic expert rules extracted from real-world tasks solved by domain experts or by ML algorithm developers. It interacted with users similarly to expert systems through question-answering sessions, in which the user was asked to provide information about the data (e.g. the number of classes or whether it could be expected to be noisy) and the desired output (e.g. rules or a decision tree). The rules were used to rank each algorithm. After the user selected an algorithm, the system executed it on the provided data and then engaged the user in a new question-answering session to assess whether the user was satisfied with the results. If not, the system would generate a list with possible parameter recommendations, which were again scored according to the stored heuristic rules.

The rules were based on user preferences and on important meta-features of the data (e.g. size, noise), on algorithms (e.g. relative memory usage, CPU time), and on the produced models (e.g. the number of decision tree nodes or the average center distance of clusters). Further, these rules triggered actions such as adjusting algorithm scores and proposing parameter settings or data transformations [Sleeman et al., 1995].

2.2.2.2. Meta-learning systems

In this section we shortly introduce the concept of meta-learning and describe the systems which provide support for users and are close to IDAs. Other Meta-L approaches are discussed in more details in Section 2.4 in terms of how ranking of algorithms is achieved.

Meta-L was born from the need of recommending which algorithms perform better for which kind of data. As more and more ML algorithms appeared recommending good algorithms for a dataset is an important step ahead. This *algorithm selection problem* was introduced by Rice [Rice, 1976] as the challenge of finding relationships between the characteristics of datasets and the performance of different algorithms.

It consists of a learning and a testing step. First, a set of algorithms is run on different datasets and their performance stored in a Meta-L case-base. Then, characteristics of datasets are extracted and a model is built. Having a new dataset and the model, in the testing phase, one can predict which algorithm works best given the data characteristics.

StatLog The StatLog project [Michie et al., 1994] was the first large-scale Meta-L approach using 20 classification algorithms on about 20 datasets. For each dataset they have identified around 20 features (simple, statistical and information-theoretic). During the training step, the system marked algorithms as *applicable* or *non-applicable*, based on how close they were to the best algorithm on a given dataset. Then, a decision tree model was built to predict if a certain algorithm is applicable or not on the given dataset. The output was a set of rules which had to be checked manually.

Data Mining Assistant (DMA) DMA [Giraud-Carrier, 2005], the outcome of the MetaL project, took the idea from StatLog one step further and automated the process. Their main purpose is to reduce the experimentation time required to choose the best classification algorithm for a given dataset. DMA uses the DCT tool [Lindner and Studer, 1999a] to extract features of the datasets. The system ranks 10 classification algorithms using a k-NN meta-learner based on their accuracy and training time (k = 3). DMA is a Web-based assistant that uses a wizard-like interface to support the user during the selection process. First, the new dataset needs to be uploaded, its characteristics are automatically computed and then the rankings are produced. For the last step, the user needs to select the ranking strategy and the trade-off between accuracy and time. DMA includes also implementations of the base algorithms such that they can be tested once selected. DMA used a set of 7 numerical data characteristics, 10 classification algorithms, and was initialized with 67 datasets, which originate mostly from the UCI repository[5].

NOEMON focuses on meta-features that are likely to give better recommendation results for classification tasks [Kalousis and Theoharis, 1999, Kalousis and Hilario, 2001a]. For this purpose, they built a meta-database which stores the performance of every two algorithms. The relevant meta-features are chosen based on feature-selection. Statistical tests are employed to decide in which situation one algorithm significantly outperforms the other one. A decision tree learner is used to build a

[5] http://archive.ics.uci.edu/ml/

model for predicting if one algorithm is better or not on a new dataset. The final rankings are computed by collecting all the models for every algorithm and counting the number of wins/ties/losses against all the other algorithms.

2.2.2.3. Case-based reasoning systems

Similarly to Meta-L systems, Case-Based Reasoning (CBR) systems use a set of experiments that are stored in a *case-base*. However, they do not learn a model to make the recommendations but rely only on the case-base. Given a new problem the system retrieves the most similar workflows according to the similarity to the new problem and their performance. Performance is most often defined by domain experts or by previous users. Having a set of options the user can load and browse them into an editor. Some systems may allow users to modify the cases, adapt them to their needs and execute them for the current problem. Once finished, the users may decide to upload their newly designed cases to the case-base. Domain experts are responsible for the case-base by designing cases or by verifying the correctness of cases uploaded by other users.

CITRUS CITRUS [Engels, 1996, Wirth et al., 1997b] was built as an advisory component for Clementine [Grimmer, 1996] – a well-known KDD suite, now part of IBM SPSS Modeler [6]. Experts have developed a case-base containing both KD operators ('processes') and streams/workflows ('sequences of processes'). Operators are described with pre- and post-conditions. When using the system, the first step is to provide an abstract task description. Also the system extracts a set of data statistics from the provided dataset. Next, CITRUS uses the CBR module to load the most similar cases in the workflow editor provided by Clementine. These, can be selected and further edited/changed. Then, CITRUS checks the correctness of the adapted workflows (based on pre- and post-conditions of operators) and removed operators that violate some of the constraints. The system uses the concept of task and task decompositions that allows to structure workflows based on tasks and sub-tasks. For this purpose CITRUS uses an hierarchical planner. [7]

AST AST [Lindner and Studer, 1999b] is a case-based reasoning system that is also based on the DCT tool. Similarly to DMA, the system returns a single algorithm

[6] `https://www-01.ibm.com/software/analytics/spss/products/modeler/`

[7] Since it employs AI planning CITRUS is also a planning-based system but its most important feature is the case-base.

rather than a complete workflow. The problem description consists first of the requirements specified by the user and then the data characteristics computed by DCT. The systems also allows incomplete problem specifications. Based on the problem description the most similar cases are computed. The system also returns results of the applied algorithms. The case base contains more than 1600 cases from 21 classification algorithms and more than 80 datasets from UCI and from real world applications.

MiningMart Likewise, the MiningMart project [Morik and Scholz, 2004] reuses successful workflows focusing more on data selection and preprocessing phases (see Fig. 2.1). It also incorporates an elaborate workflow editor that enables users to *map* a case directly to the SQL databases where the business data is stored, rather than expecting the user to first select and export the data in a format that the CBR system understands. MiningMart's online case base[8] stores workflows described in a specific XML-based language named M4. This language describes all details of the mapping process, as well as pre- and post-conditions and characteristics of all operators, which help verifying the validity of edited workflows. Instead of using a conventional CBR reasoner, Mining Mart describes cases in an ontology with informal annotations, such as the goals and constraints of each problem. The user can search the ontology and select a case. For mapping the case to the data stored in a database, it offers a three-tier graphical editor. First, in the *case editor*, the workflow can be adapted by adding or removing operators. All data in the workflow is described using abstract concepts and relations, such as 'customer', 'product', and the relation 'buys', which can each have a range of properties, such as 'name' and 'address'. Second, in the *concept editor*, these concepts can be edited to match the new problem (e.g., the property 'age' can be added to the customer). Also at this level, *concept operators* are defined, such as selecting or adding a property. Third, in the *relation editor*, concepts, relations and properties have to be mapped to tables, columns, or sets of columns in the database. To run the workflow, MiningMart translates all steps into SQL queries and operator calls, and stores the preprocessed data in the database itself. At any point, the user can access these three editors to further refine the workflow.

Hybrid Data Mining Assistant (HDMA) HDMA is a system based on ontologies and CBR to provide guidance for KDD workflows [Charest et al., 2008]. Its workflow editor offers additional guidance based on expert rules (expressed in SWRL [9]) . The

[8] MiningMart's online case-base: http://mmart.cs.uni-dortmund.de/caseBase/index.html

[9] `http://www.w3.org/Submissions/SWRL/`

ontology contains the operators and rules about their applicability. A case-base contains information about the data it was applied on (30 attributes), characteristics of the workflow (31 attributes) and ratings about the user satisfaction (5 ratings). For a new dataset and problem the system works as follows: First, the meta-features are computed based on the data. Then, the CBR system returns two scores for each case: one based on similarity and the other based on the user ratings. After having selected a case the user will be guided through the KDD workflow steps. For each step the user gets its corresponding operators and a rule reasoner further displays recommendations, which may encourage or advise against an operator based on the expert rules.

A similar approach to HDMA is the OLA assistant [Choinski and Chudziak, 2009] that employs ontologies combined with CBR focusing mainly on the collaboration between domain and technology experts. For this purpose both domain and DM ontologies are used, the later even aligned along the CRISP-DM model. Opposed to other CBR approaches OLA is not limited to the DM step, but it follows the CRISP-DM phases. It also uses the rules stored in ontologies and a description of operators in terms of inputs, output, preconditions and effects (IOPE) to dynamically compose, rank and present to the user valid processes. Unfortunately there are many missing details about this approach. There is no public knowledge if it was implemented/finished, therefore we did not add it in our comparison section.

2.2.2.4. Planning-based Data Analysis Systems

All previous categories of systems are not able to provide effective support for new problems other than just some generic advice. A way to solve this issue is to view the KDD workflow design as a *planning* problem where a plan is a sequence of operators that transforms the initial data into models or predictions. The DM operators are treated as planning operators and each of them has IOPE described in a similar manner as for planning. Then an initial problem and task are provided and then based on the planner one or more solutions are computed. We present the systems that make use of planning as their main feature and shortly introduce how they work.

AIDE AIDE [Amant and Cohen, 1998a] is a mixed-initiative planning system designed to help users during exploratory data analysis. In mixed-initiative systems, the system starts a dialogue with the user: it gives recommendations about which operators to use next and in turn allows the user to review and override these recommendations. For planning, it uses an approach similar to HTN planning [Erol, 1996], in

which the dependencies between actions can be given in the form of networks. This allows to plan on a higher, more abstract level, and to decompose a planning problem into smaller sub-problems with sub-goals. The system thus builds high-level plans consisting of sub-problems as well as low-level plans in which each sub-problem is solved by workflows of primitive operators. Moreover, AIDE also resembles CBR systems: it has a plan library (a case-base) that is used to extract plans for similar cases. These plans can also consist of sub-problems rather than primitive operators. Its planner is script-based and offers 3 primitive operations: reduction, transformation, and decomposition, which are used for data manipulation in the exploratory process.

The Intelligent Discovery Electronic Assistant (IDEA) A first PDA system for KDD is the Intelligent Discovery Electronic Assistant (IDEA) [Bernstein et al., 2005]. It regards preprocessing, modeling, and post-processing techniques as operators and returns *all* valid plans for the given problem. The IOPE information is encoded in an ontology, which also contains manually defined heuristics (e.g., the relative speed of an algorithm).

Before the planning process, the user is asked to give weights to a number of heuristic functions, such as model comprehensibility, accuracy, and speed. After planning, the workflows are passed on to a heuristic ranker, which uses the heuristics stored in the ontology to compute a score aligned to the user's objectives (e.g., building a decision tree as fast as possible). Finally, based on this ranking the user may select a number of processes to be executed on the provided data. After the execution of a plan, the user is allowed to review the results and alter the weights to obtain alternative rankings. For instance, the user might sacrifice some speed in order to obtain a more accurate model. Finally, if useful partial workflows have been discovered, the system also allows extending the ontology by adding them as new operators.

NExT NExT, the Next generation Experiment Toolbox [Bernstein and Daenzer, 2007], is a CBR-extension of the IDEA approach. It contains a case base of past workflows and uses CBR to retrieve the most useful ones. Often, only parts of these workflows are useful, leaving gaps that need to be filled with other operators. For instance, a workflow may start with a preprocessor that cannot be applied on the given data. This is where the planning component comes in: based on pre-conditions and effects of operators finds new sequences of operators to fill in those gaps. NExT includes an ontology of possible problems related to workflow execution and matching resolution strategies. Calling a planner is one such strategy; other strategies may entail

removing operators, or even alerting the user to fix the problem manually. Although evaluations of the NExT system remain scarce, its reuse of prior workflows and semi-automatic adaption of these workflows to new problems seems promising.

RDM The RDM system [Žáková et al., 2010] uses an OWL-DL-based [Patel-Schneider et al., 2004] ontology for knowledge discovery. It is organized around the concepts of knowledge (datasets, constraints, background knowledge, rules, etc.), algorithms (inductive algorithms and data transformations), and the knowledge discovery task. This ontology is queried using the Pellet reasoner [Sirin et al., 2007] and SPARQL-DL queries [Sirin and Parsia, 2007], for retrieving the operator inputs and outputs, which are then fed into the planner. Two planners are implemented. The first one is a standard PDDL planner, thus the inputs and outputs are first translated into PDDL before the actual planning phase. The second is a modification of the heuristic Fast-Forward (FF) planning system [Hoffmann and Nebel, 2001]. Here, the ontology is queried directly during the planning process. Additionally, it uses a heuristic to guide the planning: given the current partial plan, the final length of the plan is estimated using a secondary planner (GRAPHPLAN [Blum and Furst, 1997]).

KDDVM The KDD Virtual Mart (KDDVM) system [Diamantini et al., 2009b] is based on the KDDONTO ontology, which likewise describes the algorithms with their inputs and outputs as well as their properties and relationships. Akin to RDM, KDDVM interacts with the ontology by using the Pellet reasoner and SPARQL-DL queries. Instead of applying a standard planning algorithm though, it utilizes a custom algorithm that starts at the goal state and iteratively adds operators, thus forming a directed graph until the first operator is compatible with the given dataset. Operators are added using an *algorithm matching* procedure, which checks the compatibility of inputs and outputs. However, the operator interfaces do not need to be perfectly equivalent: their *similarity* is computed through their distance in the ontology graph. Finally, the produced workflows are ranked based on the similarity scores as well as other operator properties stored in the ontology (e.g. soft preconditions or computational complexity estimates).

2.2.2.5. Workflow Composition Environments

The final category of systems do not automatically propose operators or workflows, but rather support the user during manual workflow composition by offering a graphical environment, where the data flow can be drawn on a canvas, or a high-level

scripting language for quick workflow design and execution.

WCEs are the baseline of this survey since they are the only ones actively used, mainly because they focus on ease-of-use and actually executing the workflows. They are not only a collection of algorithms, but they also offer some guidance (e.g., auto-wiring, meta-data propagation, correctness checking before execution, and operator recommendations, etc.) and therefore can be considered 'pseudo-IDAs'. In the following we shortly describe the main contributors in the KDD world.[10].

Canvas-based tools *IBM SPSS Modeler*[11] is a commercial DM tool that allows the user to easily design DM workflows via an intuitive graphical interface. It contains many DM operators and provides support for all KDD steps. It also incorporates the statistic capabilities of SPSS-like data transformation, hypothesis testing, and reporting capabilities. In addition, it has advanced analytical functions, automated data preparation, and rich, interactive visualization capabilities.

SAS Enterprise Miner [Cerrito, 2007] uses the SAS methodology, SEMMA [12], designed to handle large data sets in preparation for subsequent data analysis. It provides extensive support for statistics, numerous types of charts and plots, and data exploration via hierarchical market baskets and multivariate graphical data exploration. A new feature is the SAS Rapid Predictive Modeler that offers the user an automatic guidance through the data preparation and all other DM tasks.

Weka [Hall et al., 2009] is the first open-source suite of machine learning algorithms for DM tasks. It provides support for several KDD steps like preprocessing, feature selection, DM step, evaluation, and visualization of results. Moreover, people have developed a various packages for additional tasks (e.g., tree visualization). Recently, the suite has even been integrated in the newer DM tools, like RapidMiner, KNIME, etc.

RapidMiner [Mierswa et al., 2006] is the most popular open-source system for DM.[13] It has both a GUI mode and a command-line server mode. It can be easily extended and offers many available extensions. It encompasses more than 500 operators for data integration and transformation, DM, evaluation, and visualization tasks. Moreover, it provides powerful high-dimensional plotting facilities.

[10]There are many KDD tools around, however, we limited ourselves to those we considered more relevant for the purpose of IDAs

[11]`http://www-01.ibm.com/software/analytics/spss/products/modeler/`

[12]`http://www.sas.com/offices/europe/uk/technologies/analytics/datamining/miner/semma.html`

[13]`http://www.kdnuggets.com/polls/2011/tools-analytics-data-mining.html`

KNIME [Berthold et al., 2009] is a younger system that enables the user to easily design, execute, and investigate DM workflows as well as integrate new DM algorithms. It incorporates over 100 operators for reading and exporting data (input/output), preprocessing and cleaning, modeling, analysis and DM as well as several interactive data visualizations. KNIME is compatible with Weka and the R framework via plug-ins.

Orange [Demšar et al., 2004] is another large toolbox for ML and DM routines and algorithms. It offers more than 100 operators and covers most of the standard data analysis tasks. It further provides many different ways of visualizing data, and offers support for scripting new algorithms using Python. As it does not have any important feature of IDAs we do not include it in the comparison section.

Scripting-based tools Another important category are the script-based mathematical tools which originally did not focus on DM but rather on mathematical and visualization functions that also allow to implement DM algorithms. Later, they have extended their field and include most of the KDD operators.

MATLAB [MathWorks, 2004] is a high-level language and framework that enables users to quickly develop algorithms, analyze data, and visualize it. Its language can also be used to define workflows as small programs. This tool is mainly used for all sorts of basic data analysis, like basic statistics, matrix processing and curve fitting. Additionally it contains many extensions to more advanced analysis techniques, such as computer vision. It further supports a couple of graphical interfaces, which simplify the interaction with the toolbox. For example, Gait-CAD [14] is designed for the visualization and analysis of time series and features with a special focus on DM problems including classification, regression, and clustering.

R [Ihaka and Gentleman, 1996] is a framework for data analysis based on the high level language R and used to provide statistical functionality. As such, it resembles MATLAB, but instead focuses on data analysis: it has various statistical and graphical techniques (e.g., linear and nonlinear modeling, classical statistical tests, etc.). Many scientists prefer it because it is very extensible: new algorithms and techniques can be easily added. Further, various GUIs have been developed for it from which the most known is Rattle [15]. Rattle features several windows/tabs for visualizing data and results. Another GUI for R that supports interactive graphics is RStudio [16].

[14] http://www.iai.fzk.de/www-extern/index.php?id=656&L=1
[15] http://rattle.togaware.com/
[16] http://rstudio.org/

2.2.2.6. Other statistical analysis tools

Note that this discussion excludes some popular tools for statistical analysis, such as IBM SPSS Statistics [Levesque, 2005], Microsoft Excel [Levine et al., 1999], Data Plot [Heckert and Filliben, 2003], and DataDesk [Theus, 1998]. These tools offer an extensive sets of features for statistical data analysis as well as graphical interfaces (e.g. pull-down menus, visualization features, etc.), however their support for complex workflow design or verification is rather limited.

2.2.3. Comparison of existing systems

This section presents a comparison of various IDAs from a new perspective by considering the kind of support they provide. The purpose of this comparison is to organize the existing IDAs along a set of dimensions, to identify trends as well as their assets and drawbacks. Table 2.1 summarizes the findings on the kind of support the systems offer, their status, and availability as well as relevant references for each of them in the last columns. The user support can be divided into two main categories: first, the workflow-based support, like the first six entries of the table; second, the user-based support. In our discussion we group the types of support that have contradictory/opposite purpose (e.g., single step vs. multi-step, automatic generation vs. manual, etc.) in order to better emphasize their contributions.

2.2.3.1. Modeling a single-step vs. multi-step from the KDD-process

A first distinction of IDAs is based on the complexity of the DM advice offered. Here we distinguish between systems that offer advice on one step of the KDD-process and the ones that recommend all of them. An ideal IDA should include both types: the first helps to decide when to use a certain operator and the second what operators should be used in a sequence.

For instance, ES provide support for a single step of the KDD-process. The advice is based on the information provided by the user combined with the expert rules from the knowledge base. Most often the advice is guided by questions addressed to the user and concerns both the modeling and the evaluation steps. Consultant-2 is the most advanced ES system. It suggests new methods in case the one applied has produced unsatisfactory results. Thus, a step from the KDD-process is not seen as a single-step but as a cyclic process, where the user can reapply other algorithms in case the current results are not satisfactory. MLS systems have a similar focus: they recommend the most appropriate algorithm for a certain data set–actually, they focus

Category	System name	KDD single step support	KDD multi-step support	Graphical workflow editing	Automatic workflow generation	Workflow checking/repair	Re-use past experiences	Task decomposition	Design support	Explanations	Experimental	Analytical	DM experience level	Status/Availability	References
ES	SES	++	−	−−	−−	−−	−−	−−	−−	++	−	++	naive (REX), experienced	NA/O	[Gale, 1986], [Raes, 1992], [Hand, 1987, Hand, 1990]
	MLT Consultant	++	−	−−	−−	−−	−−	−−	−−	++	++	++	all	NA/O	[Sleeman et al., 1995]
MLS	StatLog	++	−	−	−	−−	−	−	−	−	−	++	unspecified	-/O	[Michie et al., 1994]
	DMA	++	−	−	−	−−	−	−	−	−	+	++	experienced	A/O	[Giraud-Carrier, 2005]
	NOEMON	+	−	−−	−−	−−	−−	−−	−	−	+	+	unspecified	NA/O	[Kalousis and Hilario, 2001b]
CBR	CITRUS	−	++	−	+	−−	++	++	−	++	+	−	all	NA/O	[Engels et al., 1997]
	AST	+	−	−	−−	−−	++	−−	−	−	+	−	unspecified	NA/-	[Lindner and Studer, 1999b]
	MiningMart	0	0	+	−	−−	++	−	−	−	++	−	experienced	A/O	[Morik and Scholz, 2004]
	HDMA	0	0	−	−	−−	++	−	−	+	+	−	all	NA/U	[Charest et al., 2008]
PDAS	IDEA	−	++	−	++	−−	−	−	−	−	+	−	intermediary	NA/O	[Bernstein et al., 2005]
	RDM	−	++	−	++	−−	−	−	−	−	+	−	intermediary	NA/U	[Žáková et al., 2010]
	KDDVM	−	++	−	+	−−	−	−	−	−	+	−	intermediary	A/U	[Diamantini et al., 2009b]
	eIDA	+	++	+	++	++	+	++	+	−	+	−	intermediary	A/U	[**?**]
	NeXT	0	+	+	+	−−	+	−	+	−	+	−	intermediary	NA/O	[Bernstein and Daenzer, 2007]
	AIDE	+	+	−	+	−−	+	+	−	+	+	−	intermediary	NA/O	[Amant and Cohen, 1998b]
WCE	IBM SPSS Statistics	0	0	−	−	−−	−	−	−	+	++	−	experienced	A/U	[Levesque, 2005]
	R	0	0	−	−	−	−−	−	−	−	++	−	experienced	A/U	[Ihaka and Gentleman, 1996]
	MATLAB	0	0	−	−	−−	−	−	−	−	++	−	experienced	A/U	[MathWorks, 2004]
	IBM SPSS Modeler	0	0	++	−	++	−	−	++	++	++	−	experienced	A/U	Link to project site[1]
	SAS Enterprise Miner	0	0	++	−	−−	−	−	++	++	++	−	experienced	A/U	[Cerrito, 2007]
	Weka	0	0	++	−	−−	−	−	+	+	++	−	experienced	A/U	[Hall et al., 2009]
	RapidMiner 5.0	0	0	++	−	++	−	−	++	+	++	−	experienced	A/U	[Mierswa et al., 2006]
	KNIME	0	0	++	−	−−	−	−	++	+	++	−	experienced	A/U	[Berthold et al., 2009]

[1] `http://www-01.ibm.com/software/analytics/spss/products/modeler/`

++	= well supported (a main feature of the system)		A	= publicly available
+	= supported		NA	= not publicly available
0	= neutral, the system can do it but there is no assistance		U	= up-to-date
−	= not present but integrable		O	= outdated
−−	= not possible			

Table 2.1.: Overview of IDAs by offered support

primarily on the modeling step (classification and regression).

Nevertheless, optimizing a single step is not an easy task. If we observe the modeling step, for instance, it becomes apparent that each algorithm has several parameters. For a naive user with little understanding of the DM domain, it is not trivial to set their values. Probably most of these users prefer to apply just the default values.

The shift from single-step to multi-step was initiated by the CRISP-DM standard and the CITRUS system. Both approaches guide the user through all phases of the KDD-process. WCE, on the other hand, are neutral, since they allow the users to exe-

cute single operators for which, however most often, they have to set the parameters themselves. They further enable the user to design and execute multi-step KDD-processes, but this becomes hard if the processes have a large number of operators.

Above all, PDAS and partially CBRs support multi-step recommendation. They recommend correct workflows by taking the operators' pre-conditions and effects into account. Also some of the parameters are set (e.g., size of bins, number of bins). IDEA [Bernstein et al., 2005] provides its users with systematic enumerations of valid KDD-processes and rankings by different criteria. However, the system does not support the user through the steps of the KDD-process since it merely enumerates all steps. The workflow composition involves choosing induction algorithm(s) and appropriate pre- and post-processing modules. Though, it is limited to only a few operators for each step. Opposed to IDEA, AIDE [Amant and Cohen, 1998b] does not provide support for the overall KDD-process. For example, if you load a data set, the steps that are applied are: collecting information about each variable, the generation of statistical summaries, testing of the data types, testing of the continuous or discrete data, generation of hierarchical clusters for numerical data. Then a set of indications is generated, like indication of skew or indication of clustering tests for outliers. Further on, the user can make decisions and influence the execution of the plan generator.

In conclusion, we observe that most of the user support concerns the modeling steps or the automatic generation of all steps. However, preprocessing and feature selection steps are rarely addressed.

2.2.3.2. Graphical editing vs. automatic generation

Here we compare systems that are able to automatically generate workflows, i.e., PDAS, to systems that force the user to design the workflows manually. Ideally, these aspects should be both integrated into IDAs, automatic generation should be on demand and the graphical editing as default.

IDEA uses straightforward search for automatically outputting the valid processes. Here the user can select the plan by choosing a ranking criterion (accuracy, speed, etc.). The approach in AIDE differs from IDEA in the manner in which the user and the machine interact: AIDE offers a step-by-step guidance based on the script planner and the user's decisions. This is suitable for exploratory statistics but is not suitable for domains where algorithms run for an extensive period of time. Similarly, CITRUS combines planning with user guidance in order to help the user construct the best plan using a limited number of operations. IDEA only enumerates workflows, but it

does not allow the user to modify neither nodes nor parameters. However, AIDE is user-centric and can support modifications at any decision point. The user is even able to go back several steps and modify decisions taken by the system. Beyond that, the workflows generated by IDEA are valid since the operators are modeled in an ontology with IOPE which need to be satisfied in order to be able to apply the operator. Moreover, IDEA provides an auto-experimentation module, which can execute the generated plans and rank them by accuracy.

Similar approaches are RDM and KDDVM, which generate abstract workflows using an ontology combined with planning and reasoning. RDM is able to generate several plans, however, there is no statement about what the maximum number could be. [Žáková et al., 2010] exemplify one workflow for each of the application domains. In the evaluation section, the planner performance results are presented and four workflows are considered for each of the domains. Beyond that, RDM uses a repository for storing all the constructed workflows. When the user solves a DM task, the repository is searched in order to retrieve a solution, however only one. If no solution is found, the planner is called and several workflows are produced. The evaluation of the planner has been conducted on two application domains, namely CAD and gene data using two ontologies (KD-RDM and KD-Orange), but only the time for generating a workflow is measured. Hence, the generated workflows are not evaluated in terms of accuracy.

Most of WCEs allow the users to manually design workflows. They provide several tabs with various functions (e.g., selecting and configuring operators). The data objects are implicitly represented as inputs or outputs of operators and can be set in the operator configuration views. Additionally, there are different operators for reading/writing different data formats. The data flows into/out of the operators through ports, which are visible. Then the users can manually drag an output from an operator to become an input for the next operator. However, they do not use abstract nodes for grouping the operators (except for some degree RapidMiner and IBM SPSS Modeler). Furthermore, the workflows are plain, rather than a hierarchy of tasks. IBM SPSS Modeler has the concept of super nodes and sub-nodes. RapidMiner, on the other hand, uses a ‘Subprocess’, which contains an operator chain.

2.2.3.3. Workflow checking/repair

Both checking and repairing of workflows are important aspects of IDAs. Checking allows discovering errors at an early stage (before executing the workflow) and repairing fixes the problems discovered after the workflow execution. Therefore both save

time when analyzing data.

Unfortunately, only a few of the existing IDAs can check the workflows for correctness and suggest possible solutions to fix the erroneous ones. This is one of the features encountered in some of the WCEs or in PDAS. For example, both RapidMiner and IBM SPSS Modeler offer meta-data propagation–the characteristics of the input data are available at any step, which helps to detect problems. RapidMiner 5.0 also supports the notion of "quick-fix", i.e. it suggests possible fixes for errors in the workflow design. In case the input data has missing values and the operator is not able to handle missing values, it will complain and suggest using a 'replace missing values' operator.

PDAS use IOPEs for operators to ensure that they are used/applied in the right situations. However, they are not able to repair workflows. For example, if one workflow crashes during execution due to an error, no corrections are suggested.

Most of the existing IDAs are missing both features mainly as real users never tested them or as they do not offer support for an overall workflow (like ES, MLS), but also checking and repairing requires more complexity.

2.2.3.4. Task decomposition vs. plain plan

Structuring the domain data into tasks and sub-tasks simplifies and improves the automatic generation of KDD workflows and therefore constitutes a desirable feature of IDAs. However, only a few IDAs use this model. *Task-oriented user guidance* is implemented in CITRUS. Its user guidance module offers assistance in the selection and application of available techniques as well as the interpretation and evaluation of results. AIDE also uses hierarchical problem decomposition techniques: goals can be decomposed into several sub-goals. Moreover, problem decomposition and abstraction constitute helpful features for the exploration in AIDE.

CRISP-DM [Chapman et al., 1999] follows a hierarchical process model having a set of tasks at four levels of abstraction: phase, generic task, specialized task, and process instance. The KDD-process consists of 6 phases, each of them comprising several generic tasks that cover all the possible DM situations. The specialized tasks describe how the actions from the generic tasks should be accomplished in certain situations. The last level, namely process instance, represents a record of actions and results of a specific DM operation. Tasks represent abstractions over several operators, for example the prediction to fill missing values and the prediction to predict the target have the same task. A similar approach is the one from eIDA that uses tasks and methods to guide the generation of workflows. This in turn reduces significantly

the plan space and speeds up the plan generation.

2.2.3.5. Design support vs. explanations for result/output

IDAs, in general, provide either support for design or support for explanations. For an ideal IDA, both should be considered since they complement each other, designing an workflow is easier when useful explanations are present.

WCE enable users to manually design their workflows. Canvas-based tools include different types of design support. Besides the meta-data propagation and the quick fixes, RapidMiner also employs descriptions of operators stored in a wiki. This wiki contains information about the algorithms, how they work, and descriptions of their important parameters. SAS Enterprise Miner has integrated debugging and runtime statistics. On the other hand, scripting-based tools provide help facilities with explanations about the functions. R and Matlab by themselves do not include design support, however, their respective GUIs partially support it. HDMA offers some explanations based on the stored expert rules. IBM SPSS Statistics offers more support for *explanations* than other IDAs: the help menu provides extensive information about methods and algorithms even with examples illustrating the explained feature. Additionally, it features coaches that take you step-by-step through the process of interpreting the results or deciding which statistical analyses to choose via providing helpful examples. Hence, *learning by example* is a valuable feature of an IDA.

Opposed to WCEs, SES offer more support for explanations and interpretation of results. REX [Gale, 1986] helps the user in *interpreting intermediate and final results* and gives useful instructions about statistical concepts. Springex has a primitive “why” explanation facility, which consists of a list of rules that have succeeded together with the conditions that have been asserted. However, the knowledge is unclear since it does not provide an explanation of technical terms, it is superficial and incomplete. On the contrary Statistical Navigator uses an expert system with help and explanation facilities. Additionally, it has extensive reporting capabilities, including a short description of each technique and references to the literature and statistical packages that implement the technique. KENS and LMG provide explanations for concepts, but they do not handle the interpretation of results or explanations of the reasoning behind.

2.2.3.6. Experimental vs. analytical approach

Here we compare IDAs that can execute KDD-processes with the ones that only provide advice. Ideally an IDA should combine both execution and advice. The advantage of executing workflows is that the real performance comes from executing the

process and all design support loses if the results of support cannot be directly used. However, only enumerating and executing many processes it is not feasible when a large number of processes/operators are present.

WCEs allow executing workflows, as opposed to most ES that use underlying statistical packages for execution (e.g., REX is based on the S statistical system [Gale, 1986]). Most of the PDAS rely on other data analysis tools or on web services for executing the generated workflows (eIDA, RDM, etc.). Relying on external WCEs or statistical packages can cause problems of scalability: the underlying package may change, evolve and therefore the assistance may need to be adjusted accordingly.

2.2.3.7. DM experience level

An ideal IDA should consider different levels of user support depending on the user expertise with the DM domain. However, this is not a trivial feature and requires a more complex user support model. We defined four different categories of users: novice users – with only a little knowledge about ML algorithms, intermediary users – with a certain level of knowledge, expert users – with an extensive knowledge about the DM tasks, all – tools which address all types of users and unspecified – there is no evidence about the type of users. Most of the IDAs target expert or intermediary users. Only a few take novices into account and REX is one of these systems. However, KENS focuses on users with a certain level of knowledge. Inexperienced users can easily feel lost in SPRINGEX and Statistical Navigator since both systems offer a large amount of knowledge. Any domain expert can use MLT Consultant and CITRUS. Current WCE systems have features that support naive users like explanations and help facilities. However, building workflows requires more effort and knowledge because of the large amount of operators. Some WCEs try to improve this process by providing on-the-fly error recognition, quick fixes, workflow templates, re-usable building blocks (e.g., RapidMiner, IBM SPSS Modeler).

2.2.3.8. Status and availability

IDAs are useful tools for data analysis; therefore, they should be publicly available and maintained over the course of time. For the status we have Up-to-date (U) or Outdated (O) systems. WCEs are the most used since they offer a broad range of algorithms and are employed for analyzing real data. Most of the other IDAs exist only as a proof of concept and were not designed to be kept up to date or applied to real-world tasks. However, these outdated systems have a great historical value because they were based on good ideas that are either reused in later systems (ES,

MLS, CBR) or worth revisiting. For the availability metric, we use either (publicly) Available (A) or Not Available (NA). WCEs are all available and frequently used. Some older systems, such as DMA, MiningMart are also still available, even if they are no longer maintained, while some of the (younger) CBR and PDAS systems are not (yet) released for the public (e.g., HDMA and RDM). Even if these systems are currently not publicly available they are definitely worth exploring since they lead to better systems or emphasize an important characteristic of IDAs.

2.2.4. Final remarks

To conclude our comparison, we identify and explain the support limitations due to gaps in the systems' background knowledge – the main knowledge they use (DM steps, ontologies, rules, etc.). Considering Table 2.1, we observe that the missing features in the background knowledge can justify many missing features in the support.

SES systems are completely lacking the concept of workflow, which can be easily explained by the fact that they only consider the DM and evaluation steps. They are missing the first steps of the KDD-process. This further explains missing features, like graphical editing of workflows, automatic execution, etc. Having a set of hardcoded rules at hand and focusing only on one single task makes it easy to provide result interpretations and explanations for users.

Similarly, MLS systems focus primarily on the DM step, i.e., mostly on classification and regression problems. The single step support is correlated with the presence of a predictive model, or in the case of ES, with the presence of rules. MLS employ the models to recommend the best suited method for a certain data set and task; on the other hand, ES use rules. If the system, however, does not produce predictive models, the support for single step is missing altogether.

Being able to handle several steps of the KDD process does not necessarily mean that the system also provides support for multiple-step. For example, WCEs allow building workflows, but they do not provide sufficient guidance on the order of the operators. Table 2.1 suggests that if the IOPE are described (especially the conditions and effects) then the system is able to automatically generate workflows. Knowing when the operator can be applied and what it produces is essential for automated generation of workflows. This explains the workflow-related features of the PDAS. It is further a justification for the same missing features from the statistical and DM tools. Most of the WCEs allow multiple steps, but they have no description of operators' conditions and effects, therefore, they cannot decide when to apply an operator.

However, current WCEs are improving and try to integrate such information.

Looking at the systems' status and availability we can argue that even if systems may analytically be ideal to solve a certain problem, they still need to be integrated into a useful, up-to-date environment to be of any use for the end user. Therefore, they should be integrated into an execution framework (e.g., as a WCE extension).

2.3. Automating the KDD process

As found in our survey an IDA should be able to automatically generate all the possible correct workflows. This feature provides more support to users and brings IDAs one-step closer to the ideal IDA. This section explores in depth different approaches for automating the KDD process. Most of them combine ontologies and planning to automatically generate KDD workflows for a given problem. First, we compare these ontologies along their purpose and content. Second, we give an overview of planning systems. Last, we are interested in systems that are able to rank workflows or ML algorithms along different measures.

2.3.1. Ontologies for Data Mining and KDD

Over the last 10 years researchers have gained more and more interest in ontologies which are now available for various fields [McCuinness, 2005]. These allow to structure knowledge from different domains enabling sharing and reuse of data [Gruber, 2008].

There have been several attempts to build ontologies that incorporate the knowledge from DM or KDD. Since KDD arose from several fields (Statistics, ML, Databases, etc.), one of the challenges is to align terms and interpretations about several terms. The available KDD or DM ontologies can be grouped by their purpose as follows: generic ontologies for DM with main focus on algorithms for the data mining step, ontologies for KDD in the grid and ontologies for automatic planning of KDD workflows. In the following we present each of these categories in more detail.

DM ontologies OntoDM [Panov et al., 2008] focuses more on the DM step and was built as a generic ontology based on the general framework from DM [Džeroski, 2007]. It is designed as a heavyweight ontology that focuses on concepts and their relationships as well as their application domains. It includes also methods for mining structured data, biological and environmental data. The ontology development follows

a top-down approach and relies on the OBO Foundry design principles [17] to ensure a sound theoretical foundation. The ontology contains the following entities: dataset, data type (primitive and structured), tasks (e.g., predictive modeling, clustering, pattern discovery, etc.), generalizations (e.g., probability distributions, predictive models, clustering and patterns), algorithms, components of algorithms (e.g., distance functions, features, kernel functions, etc.) and constraints (language or evaluation).

The Data Mining Optimization Ontology (DMOP) [Hilario et al., 2011] provides a very fine-grained description of the DM tasks, algorithms, models, datasets and performance metrics. The ontology is mainly used for algorithm and model selection which in the end translates to Meta-L. The core concepts of the ontology are: *Data* (*Model*, *Report*, etc.), *DM-Experiment*, *DM-Workflow*, *DM-Operation*, *DM-Operator*, *DM-Algorithm* and *DM-Task*. They are connected through a set of relations as follows: a *DM-Experiment* executes a *DM-Workflow* that has a set of *DM-Operator*s. At its turn *DM-Operator* implements a *DM-Algorithm* that addresses a *DM-Task*. Each algorithm, task, operation and experiment has a set of inputs and outputs. The data has a set of features that are described in detail (number of instances, number of features, number of categorical features, number of classes, etc). The tasks are structured in four main categories: *DataProcessingTask* (every operation on attributes - *FeatureSelection*, *FeatureDiscretization*, *FeatureWeighting*), *ModelingTask* (*PatternDiscovery*, *DescriptiveModeling*, *PredictiveModeling*) and *ModelApplication* (*Prediction* and *Evaluation*). The algorithms are also classified in three main categories: Generative (compute class-conditional densities and the priors for each class – NaiveBayes), Discriminative (compute posteriors to determine class membership – LogisticRegression) and DiscriminativeFunc (build direct mapping from input onto class label – SVM, NeuralNetworks, etc.).

Exposé [Vanschoren et al., 2012] is a DM ontology with focus on experiments. It builds upon the basic classes for DM from OntoDM and the ones from EXPO [Soldatova and King, 2006] that contains basic concepts for experimental design. It also reuses concepts from DMOP [Hilario et al., 2011] to describe in detail each ML algorithm. The ontology extends all these three imported ontologies and adds more detailed information about experiments, algorithms, configurations, and evaluation. Several kinds of experiments are available and defined in a very fine-grained manner (e.g., learner evaluations). Experiments are similar to workflows having inputs, outputs, and sub-workflows. Another feature is ability to differentiate between different implementations of the same algorithms. Exposé is developed in OWL-DL. It

[17] `http://obofoundry.org`

also leads to a language ExpML which allows to describe experiments in XML. This language seems similar to the RapidMiner XML language for workflows.

Ontologies for distributing KDD in the grid Another approach is the DAMON ontology [Cannataro and Comito, 2003] used in the design of distributed KDD applications in the Grid architecture. The ontology is built with the purpose of having a reference model for domain users that incorporates different DM tasks, methodologies and descriptions of algorithms. The users should be able to search different elements of the KDD domain by various needs. The ontology is implemented in the ontology language DAML+OIL [18]. The main classes from the ontology are: *Task*, *Method*, *Algorithm* and *Software*. These are connected via a set of properties as follows: a *Software* implements an *Algorithm*, an *Algorithm* uses a *Method* and a *Method* specifies a *Task*.

A similar approach is the one from Universal Knowledge Grid (UKG) [Yu-hua et al., 2006]. They have also built the ontology for distributing KDD on the grid. The ontology is implemented in OWL [19] and it also integrates information about the application domain. The main classes of the ontologies are related as follows: an *Application_domain* has several *Sub_domain*s (e.g., banking, bioinformatics, direct marketing, fraud detection, e-commerce, investment, etc.). Each *Sub_domain* has several *Application_task*s. Each such task can have one or more *Solution*s which in their turn include one or more *Algorithm*s. An algorithm performs a *Function* which is own by *Data_mining*. Also, algorithms handle different *Types_of_data* that are also handled by *Data_mining*. A *Solution* also accepts a *Predict_model* expressed in PMML [20]. The ontology is exemplified on a money-laundry use-case. A distinctive feature of this ontology is the possibility to add information about the application domain. Also it seems not to include information about the distinct steps of the KDD-process.

Ontologies for automatic planning of KDD workflows One of the first KDD ontologies is used by the IDEA system [Bernstein et al., 2005] and it follows the steps from the KDD-process as described in [Fayyad et al., 1996b]. The ontology contains a taxonomy of KDD algorithms: operators for pre-processing, data mining and post-processing steps as shown in 2.4. For each operator the ontology stores inputs, outputs, conditions–when the operator can be applied and effects–what are the effects on the data, and also estimations about the effects on speed, accuracy, etc.

[18] `http://www.w3.org/TR/daml+oil-reference`
[19] `http://www.w3.org/TR/owl-guide/`
[20] `http://www.dmg.org/v3-0`

This is only a prototype ontology that contains for each step a limited number of operators from Weka. The limitation of this ontology is the fact that it contains only a few operators. Also the input and output data are not described in such details as to allow it to be used during the generation of workflows.

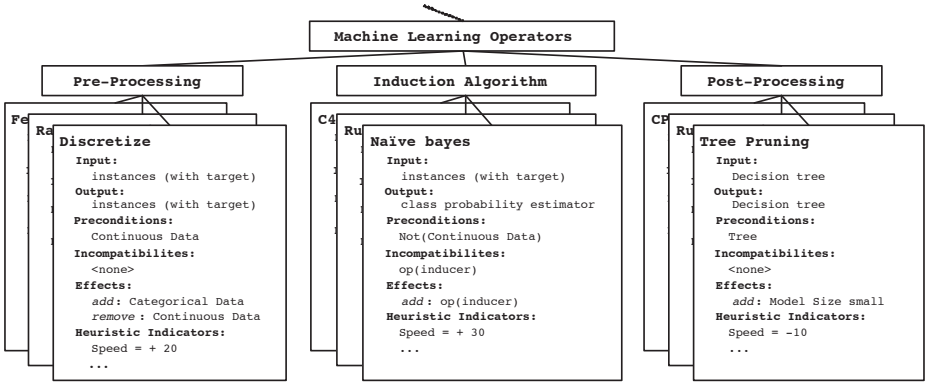


Figure 2.4.: Skeleton of IDEA ontology (taken from [Bernstein et al., 2005]).

KDDONTO [Diamantini et al., 2009a], similarly to IDEA, is an ontology with focus on both the discovery but also the KDD-process composition. The ontology is implemented in OWL-DL and is part of the KDDVM project that focuses on discovering web services for executing KDD algorithms. The ontology has at its core the *Algorithm* concept as it is the basic component of each process. An *Algorithm* uses a *Method* to extract knowledge from the input data. It also has a set of *parameters*, *pre-conditions* and *post-conditions*. Other important concepts are: phase–step of the KDD-process, task–the main purpose of the analysis, model–extracted knowledge, dataset–data in different formats. An important feature of this ontology is the flexibility of KDD phases : same algorithms can be used in several steps of KDD (as is normally the case, e.g., pre-processing (to remove/predict missing values) or data mining). The ontology contains 95 classes, 31 relations and more than 140 instances that consist of algorithms for all phases (e.g., Feature Extraction, Classification, Clustering, Evaluation and Interpretation). Each algorithm is characterized by inputs, outputs and parameters. The algorithms are designed to be compatible with OWL-S specification. The ontology has been extended to support workflow features: *KnowledgeDiscoveryTask*–with initial and goal specifications and *Workflow*–subclass of *Algorithm*, consists of a set of actions (property *hasAction*). Each action has attached an algorithm configuration (property *hasAlgorithm*) and a starting time (property *startTime*); dependencies between actions are specified via the *predecessor* property.

The Knowledge Discovery (KD) ontology [Žáková et al., 2010] captures background

knowledge about the KDD steps and the relational DM algorithms such that it can be used as a domain for planning. The main classes from the ontology are: *Knowledge*, *Algorithm* and *Task*. A specific characteristic of the KD ontology is the fact that it handles different types of knowledge: dataset (relational data), logical knowledge–with different expressivity levels and non-logical (results of inductive mining algorithms–language bias, generalizations as patterns and models). The ontology is implemented in OWL-DL and contains around 150 concepts.

2.3.2. Planning workflows/services

AI planning is used to solve more and more real-world problems. During last years researchers used planning in other areas: planning web service compositions and planning KDD workflows. These two problems are quite similar in the sense that both have the concept of processes/workflows that need to be composed. To apply planning to a problem one needs to describe the planning domain, the planning actions or operators and the initial state and goal. The systems also include modules for executing the web services during planning such that after each action its effects are identified.

Web service composition The approach of [Sirin et al., 2004] uses the SHOP2 planner [Nau et al., 2003] to plan the composition of web services. OWL-S [21] is used to describe the services under the form of a set of ontologies as it is the standard for web service automation. The ontology uses and describes web services as actions with conditions and effects. SHOP2, a domain-independent HTN planning system, plans the tasks in the same order in which they should be executed. It uses a language similar to PDDL [22] to describe the domain. HTN planning fits well the domain of web services: here we have also composite processes and some methods need to be reused. They provide an OWL-S to SHOP2 translator as well as an SHOP2 to OWL-S for the translation of the planned sequence of web services.

The NeXT system [Bernstein and Daenzer, 2007] uses mixed-initiative planning to compose web services. The user assists the system with suggestions, and the system in its turn guides the user to decide at any step which action is better. The system allows a flexible execution of workflows (partial execution) as well as the adaptation/change of the workflows to the user needs (parameter values of actions can be changed). Processes are described by their IOPE and can be easily used for plan-

[21] http://www.w3.org/Submission/OWL-S/

[22] http://zeus.ing.unibs.it/ipc-5/pddl.html

ning. CBR is used to find similar processes for a given task. A plug-in interface is available to connect/use different planners.

KDD workflow planning Planning has been used also to generate workflows for KDD where actions represent the KDD operators for every step of the workflow. In Section 2.2.2.4 we have introduced the main systems which follow this approach. Most of them use existing planners and the main challenge is to translate the KDD domain to the planning domain. A search algorithm is used to match the outputs from previous steps to inputs in the next ones [Bernstein et al., 2005]. A similar idea employs algorithm matching to find the right sequence of operators [Diamantini et al., 2009b]. HTN-planning was used for guiding the data analysis process focusing more on simple tasks from data analysis/statistics [Amant and Cohen, 1998a]. This is a predecessor of real PDA systems.

The limitations of existing systems are the following: their ontologies only cover a limited number of operators (IDEA, KDDVM, RDM). They also have different focuses: IDEA is only a proof of concept, KDDVM focuses on combining available services for DM and RDM plans KDD workflows for different types of data. However, none tries to plan on data tables at a very detailed level to allow refined plans.

2.3.3. Ranking KDD processes

As AI planners allow generating automatically all possible workflows for a given KDD task and dataset, the question is how to identify which workflows are better than others. This can be translated into a ranking problem. here, we shortly introduced two systems that make use of ranking for KDD workflows in terms of different performance measures.

IDEA From PDAs only IDEA takes into consideration this problem and includes a strategy for ranking the workflows based on several metrics: accuracy and time. The system provides two types of rankings: static and dynamic. For each operator the ontology provides an estimation of its effects under the form of a score. Then for the overall workflow the score of all its operators is considered (e.g., time). They show that ranking by speed can provide good results. However, ranking by accuracy is most difficult. Auto-experimentation is used to estimate the performance of a set of workflows on a given problem on the fly. A set of experiments is performed using cross-validation on a subset of the dataset to get an estimation of the accuracy of workflows. The results show that auto-experimentation can significantly improve the

rankings by accuracy, but at least 20-30% of the data needs to be used as a sample. The larger the sample the higher is the accuracy obtained meaning that one could trade higher accuracy for higher response time.

Meta-mining approach [Hilario et al., 2011] focuses on ranking KDD workflows produced by eProPlan [Kietz et al., 2012] (the same planner as the one used in this thesis). The ranking is done from a Meta-L perspective. Here, a set of experiments is performed on different datasets and a case-base is built containing the performances of operators (feature selection + learning operator) as well as a set of meta-features. They try to find which combinations of feature selection and learning algorithms give better results. For this purpose they test two scenarios: first one uses only meta-features – features of datasets as predictors. The second one combines both features of the data and workflow characteristics and provides better results. It is based on a hybrid similarity measure that uses ratings of workflows combined with meta-features [Nguyen et al., 2012]. This allows to provide recommendations when few ratings are available for a new datasets (cold-start problem).

As we know this approach uses only two-steps of the KDD workflow, however other preprocessing steps may be very important for the overall performance of the workflow (e.g., preprocessing - cleaning missing values, normalization, etc.).

2.4. Selecting ML algorithms

In this section, we shortly introduce several methods of selecting ML algorithms. This is relevant related work on ranking KDD workflows. The algorithm selection problem introduced first by Rice [Rice, 1975] has been extensively experimented and studied in several fields of research, in particular ML and Constraint Programming (CP). The approaches in these two domains rely on two main strategies: Meta-L and Meta-Optimization [Hoos, 2012]. The strategies depend mainly on a set of descriptive features, called m-features, which capture characteristics of problem instances and a dataset. Both methods use the dataset to build a meta-model that predicts how a portfolio algorithm will behave on a particular problem instance.

2.4.1. Meta-learning

Meta-L was shortly introduced in section 2.2.2 together with a set of systems that provide user support. In this section, we focus on the more general purpose of Meta-L:

recommending and ranking algorithms. This includes approaches that only rank algorithms and do not provide any additional support for users.

Several surveys addressing the Meta-L topic are available [Vilalta and Drissi, 2002, Smith-Miles, 2008], it is beyond the scope of this thesis to present an extensive view. However, we select the approaches relevant to our workflow-ranking scheme. Ranking workflows is similar to ranking algorithms and it is an interesting related field to our work.

Meta-L focuses mainly on supervised learning algorithms (e.g., classification and regression). The best suitable algorithm for a given dataset is predicted by learning the relationship between the characteristics of datasets and the algorithm performance. It has two stages (training and testing), where the training phase first collects a set of features for datasets which greatly influences the performance of the meta-learning approach, produces a meta-base with executions of several ML algorithms and generates a model that maps data characteristics to performance of algorithms. In the testing phase, a new dataset is tested and ranks for the algorithms are predicted.

To compare algorithms accurately, NOEMON [Kalousis and Theoharis, 1999] uses pairwise comparison of every two algorithms. The quality of features is improved by applying automatic feature selection [Kalousis and Hilario, 2001a]. A decision tree learner builds the prediction model that decides when one algorithm is better than another one. Improvement on previous approaches on Meta-L have been done by using different algorithms for building the model. Some of the interesting approaches use boosted decision trees [Kalousis, 2002], predictive clustering trees [Todorovski et al., 2002], regression algorithms [Bensusan and Kalousis, 2001] and neural networks [Castiello and Fanelli, 2005]. Some introduce new implementation frameworks, such as METALA [Botia et al., 2001, Hernansaez et al., 2004] and [Grabczewski and Jankowski, 2007]. A very complete overview of these systems can be found in [Vanschoren, 2010].

Landmarking is a Meta-L technique that uses the performance of some fast running algorithms on the data at hand as meta-feature. This is actually very similar to auto-experimentation but is done on the algorithm level. A set of basic algorithms is considered as landmarkers. Some new meta-features for problems based on the concept of landmarking were proposed by [Pfahringer et al., 2000]. The approach tries to locate the new problem in the problem space based on its execution on some simple and efficient learning algorithms and seems to get comparable results to normal meta-learning. Subsampling landmarking [Fürnkranz and Petrak, 2001] applies

more complex algorithms on subsets of the data to avoid applying only computationally simple algorithms. However, this does not improve upon previous results. A more recent approach [Leite and Brazdil, 2010] explores a combination of landmarking with classic Meta-L and shows a better performance than previous approaches.

2.4.2. Constraint Programming

Algorithm selection is an important problem in the CP research field [Rice, 1975]. For this purpose several algorithmic frameworks have been developed.

Combining different solvers – so called portfolio solving – provides better results than just applying an individual solver [O'Mahony et al., 2008, Xu et al., 2008]. CPHydra [O'Mahony et al., 2008] uses a case-base of solved problems to solve new unseen problems. For each problem the system stores in the case-base features of the CSP problem instances and the solver times. CBR is used for selecting the right solver and portfolio generation as it has been proven successful in solving weak-theory problems from complex domains.

Satzilla [Xu et al., 2008] uses a similar concept of algorithm-portfolio, but instead of using a case-base it builds empirical hardness models using ML techniques (ridge regression) to predict the runtime of an algorithm on a given problem instance. The approach has two stages: First, an algorithm portfolio is built offline. Having selected some training instances, a set of pre-solvers is applied on easy instances to ensure that empirical hardness models focus on the hard instances. Then for each problem instance relevant features are identified and each algorithm is executed to provide estimates of its running time. A validation dataset then establishes which solver achieves the best performance for all instances. An empirical hardness model is built for each algorithm that predicts the runtime based on the features. The second stage is dedicated to solving a given problem instance online. Pre-solvers are run until a certain time, then feature values are computed until timeout. The backup solver is used when timeout occurs. Otherwise, the runtime is predicted using the hardness models and then the best algorithm is run, if that fails then the next best one is selected.

Similarly, ParamILS [Hutter et al., 2012, Hutter et al., 2012] uses an optimization approach to iteratively determine the optimal parameter setting for a given instance and algorithm.

2.4.3. Limitations

All the presented approaches rely on three main components: i) a set of meta-features mapping every problem instance into a vector $\mathbb{R}^d$, ii) a meta-dataset associating the target algorithm performance indicators (runtime for CP, accuracy, regression error for ML) to the meta-representation, and iii) a learning or optimization algorithm for algorithm selection and hyper-parameter tuning.

The approaches from both ML and CP are converging to similar techniques (CBR, features as Meta-L, etc) as the purpose is relatively similar. CP approaches can recommend good solvers by experimenting, which is partially done also by meta-learning especially by landmarking. In CP the focus is mainly on building and exploring the meta-dataset. In contrast, in ML the main effort is invested in the definition and selection of relevant m-features. The main issue is that the most informative features are computationally expensive. Additionally, most of the research in ML was done for algorithms not for whole KDD workflows. Meta-L has mainly explored the performance of classification and regression algorithms on UCI datasets.

2.5. Discussion

This chapter presents the main concepts from relevant areas of research and related work. An important part is the survey about existing IDAs which allows to identify a set of features that IDAs should possess. This explores how the KDD process should be improved to offer better support to users during their data analysis process.This chapter has also addressed RQ1 and focused on the first two hypotheses. The large number of available techniques that increases every year poses problems to users when analyzing their data. The survey of existing research has proven that KDD tools lack many features from which users could benefit and which would transform them into real IDAs.

Driven by the complexity of the KDD-process and by the large number of available techniques, the focus needs to change from algorithms to workflows. Automating the KDD-process is a major need for future data analysis tools since existing approaches do not provide scalable solutions. There is no KDD tool that already incorporates this feature. Based on our survey, identified features and comparison of existing systems, we accept the first two hypotheses. Generating correct workflows solves partially the problem, however selecting a good workflow remains to be solved. These two issues are the focus of this thesis. Based on the identified gaps in the related work we present our approach on building a better IDA and providing real support for the

KDD-process.

3
Automating KDD

From the related work about current KDD tools we can clearly see they are lacking the capability of automatically generating possible correct workflows for a given dataset [Serban et al., 2012]. Previous attempts of solving this problem only scratched the surface and did not provide a sustainable solution, however they managed to prove that techniques from classic AI could be used to solve it. In order to overcome this problem we have proposed to go one step forward and provide a sustainable solution that allows to maintain and optimize the KDD domain over time. This combines ontologies and AI planning to generate all possible workflows. Additionally, we have developed a system that allows modeling, testing and maintaining the planning domain, as KDD is a very dynamic field.

3.1. Introduction

In section 2.2 we have identified a set of support features that an ideal IDA should have. One of the most important features was to be able to get correct workflows for a dataset and task without having to have in-depth knowledge about KDD and ML algorithms. We also discussed some approaches that used ontologies and AI planning to achieve this goal. In this section, we take a similar idea but make it feasible for KDD tools (in the context of RapidMiner). Moreover, we rely on an already

proved technique in AI, HTN-planning, an approach that gives good results for real-world problems. The resulted method is able to generate correct and even avoid useless combinations of operators.

3.2. Data Mining Workflow Ontology

The Data Mining WorkFlow ontology (DMWF) [Kietz et al., 2009, Kietz et al., 2010b] contains all the information about the KDD process. It was built specifically to describe the KDD domain for planning. It considers the steps of the KDD workflow as defined in the CRISP-DM standard and provides various methods or algorithms for each of them.

The ontology is expressed in OWL 2 and is a conceptual schema (called Tbox) that has classes, properties, their characteristics and individuals (instances). It also contains a set of axioms that express the relationships between concepts.

The main classes from the ontology are: Input/Output Objects (*IO-Objects*)—data used or produced by algorithms, operators—the ML algorithms, goals/tasks—problems that need to be solved and tasks/methods—how is each step of the KDD process solved. In the following sub-sections, we are discussing in details the relevant classes of the ontology since they are important to the automation of the KDD process.

3.2.1. Meta-Data of Input/Output Objects

IO-Objects are the data used or produced by algorithms and it contains different categories as shown in Fig. 3.1a: *Data* (e.g., *DataTable*, *DataCollection*, *ImageCollection*, etc.), *Background Knowledge* (*AttributeWeights*), *Model* (*PreprocessingModel*, *PredictionModel*) and *Report* (*LiftChart*, *PerformanceMatrix*). They define the data used and produced by algorithms. There are three main types of input data: *DataTable*, *Image* and *DataCollection*. In this work, we only consider the first one since most of the classic ML algorithms work on this type of data. To test the usage of the last two one needs to describe matching algorithms in a similar way.

Figure 3.1.: Sub-classes of DomainModel

IO-Objects have several characteristics that are usually found under the term of *meta-data* as shown in Fig. 3.1b. The most important input object is the *DataTable* that consists of a set of attributes or columns. The meta-data of a *DataTable* is described at such a level of detail that allows defining when operators should be applied on it (e.g., they have columns/attributes of different types with or without missing values). Depending on the types of data or on some statics of the data, *DataTable*s can be classified as shown in Fig. 3.2.

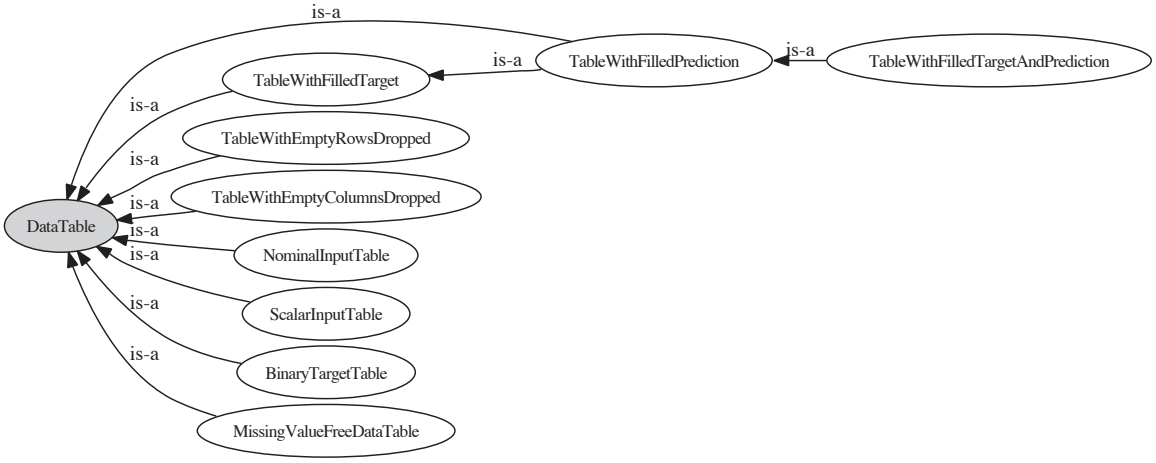


Figure 3.2.: Types of DataTables

The characteristics of a dataset are extracted and stored in an ontology as ABox axioms. Table 3.1 shows the characteristics of the Labor-negotiations dataset [1] that are extracted using eProPlan. As you can see such characteristics include statistics about the type of the attributes, distribution of the data (mean, minimum, maximum, standard deviation, etc.), the role of attributes, etc. All this information is used to express the conditions and effects of operators which are essential for the planning process.

To reduce the number of attributes and their characteristics one can use groups of attributes/columns. This feature allows grouping attributes that have similar types and behavior. For example, the meta-data with column-groups for Labor-negotiations is shown in Table 3.2. For each column group only one individual is created having the same characteristics. This feature is very useful for datasets with many attributes since it reduces the complexity of the objects used during planning. Especially biological micro-array data can easily contain several thousand columns.

[1] `http://archive.ics.uci.edu/ml/datasets/Labor+Relations`

Attribute	#Attr	Type	Role	#Diff	#Miss	Values	Min	Max	Mean	Modal	Std.
contrib-to-dental-plan	1	Categorical	input	3	15	[none,full,half]				half	
contrib-to-health-plan	1	Categorical	input	3	16	[none,full,half]				full	
col-adj	1	Categorical	input	3	16	[none,tc,tcf]				none	
class	1	Categorical	target	2	0	[good,bad]				good	
pension	1	Categorical	input	3	22	[none,employ_contr, ret_allw]				none	
stand-by-pay	1	Scalar	input		33	[]	2	13	6.142		4.845
duration	1	Scalar	input		1	[]	3.0	2.0	2.10		0.753
statutory-holidays	1	Scalar	input		2	[]	9.0	15.0	11.10		1.371

Table 3.1.: Partial meta-data for the Labor-negotation dataset

Attribute	#Attr	Type	Role	#Diff	#Miss	Values	Min	Max	Mean	Modal	Std.
col-adj|contrib-to-dental-plan |contrib-to-health-plan	3	Categorial	input	3	10						
class	1	Categorical	target	2	0	[good,bad]				good	
pension	1	Categorical	input	3	22	[none,employ_contr, ret_allw]				none	
stand-by-pay	1	Scalar	input		33	[]	2	13	6.142		4.845
duration|statutory-holidays	1	Scalar	input		0	[]					

Table 3.2.: Partial meta-data with column groups for the Labor-negotation dataset

3.2.2. Operators/Algorithms

Operators are the core of the DMWF ontology. They are structured in the *Operator* class hierarchy based on their capabilities (e.g., fill missing values, classification, regression, etc.). We have different types of operators: abstract operators – umbrella-like terms for algorithms with certain focus (e.g., for discretization, for cleaning missing values, for classification, etc.), basic operators – the ML algorithms that can be actually executed (mainly RM and Weka operators) and dominating operators – loop operators (e.g., cross-validation, meta-operators, etc.). The type of operators is specified using the class *OperatorType* and the object property *hasOperatorType* as shown in Fig. 3.3. The main classes of the *Operator* class are displayed in Fig. 3.4.

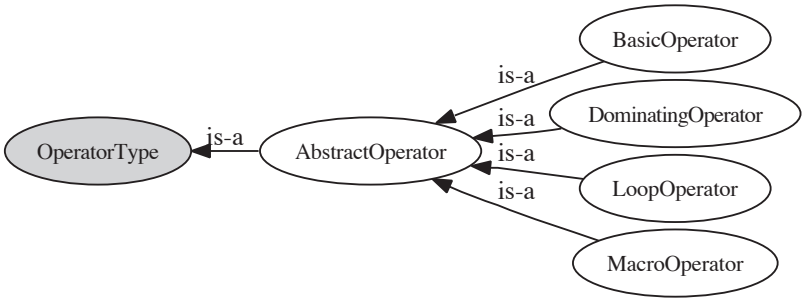


Figure 3.3.: The type of operators

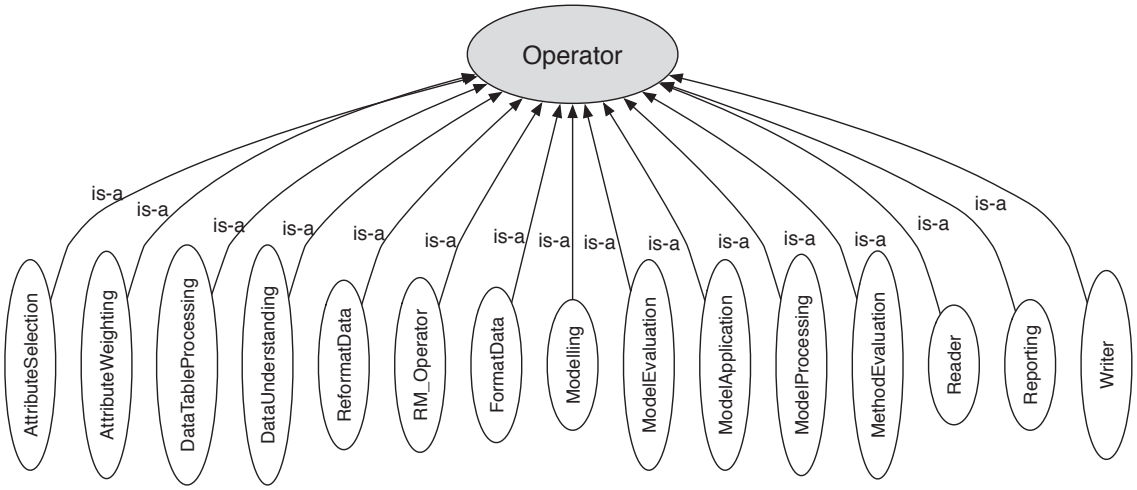


Figure 3.4.: The main classes of operators

Operators use *IO-Objects* as inputs as well as parameters to produce other *IO-Objects* (a modification of the initial object - e.g., fill missing values or a new object e.g., *Model*). In the ontology they are defined as concept expressions and stored as superclasses or equivalent classes. Conditions and effects define rules that specify when a certain operator can be applied and what it produces. The rules are expressed in SWRL [Horrocks et al., 2004] combined with concept expressions and some extended-SWRL-like built-ins. We have defined a set of built-ins to create, copy and destroy objects during planning (e.g., *copy*, *copyComplex*, *new*, *newFor*, etc.). Conditions and effects are stored as class annotations (they are more expressive than SWRL rules). Both conditions and effects are rules. Conditions check the applicability (lhs) and infer the parameter settings (rhs); different solutions can infer that the operator can be applied with different parameter settings. Effects compute the variable-bindings (lhs) for the assertion to be made (rhs); all different solutions are asserted as the effect of one operator application.

The ontology organizes operators in a hierarchy having abstract operators at the top followed by sub-classes that can be abstract, basic or dominating operators. The hierarchy allows the sub-classes to inherit the inputs/outputs/parameters (all the superclass or equivalent class definitions) from their parents/super-classes. This simplifies the modeling of operators and avoids redundant definitions. Figure 3.5 illustrates the definition of an abstract classification learner and Fig. 3.6 shows a basic operator that inherited all the *IO-Objects*, parameters, conditions and effects from its super-classes and it only has its own condition. The operator can be applied on a *DataTable* that

"RegressionLearner":

Equiv. class: *PredictiveSupervisedLearners* **and** (*uses* **exactly 1** *DataTable*) **and** (*produces* **exactly 1** *PredictionModel*) **and** (*operatorName* **max 1** Literal)

Condition: [*DataTable* **and** (*targetAttribute* **exactly 1** *Attribute*) **and** (*inputAttribute* **min 1** *Attribute*) **and** (*targetColumn* **only** (*DataColumn* **and** *columnsHasType* **only** (*Scalar*))) **and** (*inputColumn* **only** (*DataColumn* **and** *columnsHasType* **only** (*Scalar* **or** *Categorial*)))](?D) → *new*(**?this**), *ClassificationLearner*(**?this**), *uses*(**?this**,?D)

Effect: *uses*(**?this**,?D), *ClassificationLearner*(**?this**), *inputColumn*(?D,?IC),*targetColumn*(?D,?TC), → *copy*(?M,?D, {*DataTable(?D)*, *containsColumn*(?D,?_), *amountOfRows*(?D,?_)}), *produces*(**?this**,?M), *PredictionModel*(?M), *needsColumn*(?M,?IC), *predictsColumn*(?M,?TC)

Figure 3.5.: An abstract operator: RegressionLearner

"RM_Support_Vector_Machine_LibSVM_epsilon_SVR_linear":

Equiv. class: RM_Operator **and** (*usesData* **exactly 1** *DataTable*) **and** (*producesPredictionModel* **exactly 1** *PredictionModel*) **and** (*simpleParameter_kernel_type* **value** "linear") **and** (*simpleParameter_svm_type* **value** "epsilon_SVR") **and** (*operatorName* **exactly 1** {"support_vector_machine_libsvm"})

Condition: [*MissingValueFreeDataTable* **and** (*targetColumn* **exactly 1** *ScalarColumn*) **and** (*inputColumn* **min 1** *Thing*) **and** (*inputColumn* **only** (*ScalarColumn*))](?D) → *RM_Support_Vector_Machine_LibSVM_C_SVC_linear*(**?this**), *simpleParameter_svm_type*(**?this**,"epsilon-SVR"), *simpleParameter_kernel_type*(**?this**,"linear")

Figure 3.6.: A basic regression learner operator

has no missing values, its target column is scalar and all the other columns are either scalar or categorical.

3.2.3. The Task/Method decomposition

HTN-planning has been shown to be more effective for real-world planning problems [Nau et al., 1998] than classical planning. The hierarchical structure in the CRISP-DM model has a similarity to HTNs. We chose to use an HTN as it seemed a more natural approach.

In our ontology the HTN (similar to [Nau et al., 2004, Erol et al., 1995]) consists of the following: a set of goals (hints) to be achieved (sub-classes of *Goal*) and a set of tasks to solve the goals (sub-classes of *Task*). A Task has *IO-Objects* specification (using sub-properties of *worksOn*) and a set of corresponding methods (sub-classes of *Method*) that uses the same *IO-Objects* to solve the task. Each *method* has a

condition restricting its possible applications, a contribution specifying which *Goal*s it reaches, and a decomposition into a sequence of (sub)-tasks and/or operators, which executed in that order achieve the task, and finally bindings that specify the flow of the *IO-Objects* within the method (stored as annotations to the step-restrictions). Fig.

"CleanMissingValues":

Method 1: *DoneNoMVLeft*

Condition: [*inputColumn* **max 0** *NonMissingValueFreeColumn*](?Din)

Head IO: *inputData*=?Din, *inputModel*=?PM, *outputData*=?Din, *outputModel*=?PM

Method 2: *DropMVRecursive*

Condition: *DataTable*(?DIN), *inputColumn*(?DIN,?Col), *NonMissingValueFreeColumn*(?Col), *amountOfMissingValues*(?Col,?AMV), *amountOfRows*(?DIN,?AR), *divide*(?PMV,?AMV,?AR), *greaterThanOrEqual*(?PMV,0.3)

Task IO: *inputData*=?Din, *inputModel*=?PM0, *outputData*=?DOUT, *outputModel*=?PM

Step 1: *RM_Select_Attributes_subset*

Step IO: *uses*=?DIN, *produces*=?DINTER, *parameter_column*=?Col

Step 2: *CleanMissingValues*

Step IO: *inputData*=?DINTER, *inputModel*=?PM0, *outputData*=?DOUT, *outputModel*=?PM

Method 3: *FillMVRecursive*

Condition: *DataTable*(?DIN), *inputColumn*(?DIN,?Col), *NonMissingValueFreeColumn*(?Col), *amountOfMissingValues*(?Col,?AMV), *amountOfRows*(?DIN,?AR), *divide*(?PMV,?AMV,?AR), *lessThanOrEqual*(?PMV,0.5)

Task IO: *inputData*=?Din, *inputModel*=?PM0, *outputData*=?DOUT, *outputModel*=?PM

Step 1: *RM_Replace_Missing_Values_subset*

Step IO: *uses*=?DIN, *producesData*=?DINTER, *producesPrePropModel*=?PM1, *parameter_column*=?Col

Step 2: *RM_Group_Models_append*

Step IO: *usesFirstModel*=?PM0, *usesSecondModel*=?PM1, *produces*=?PM2

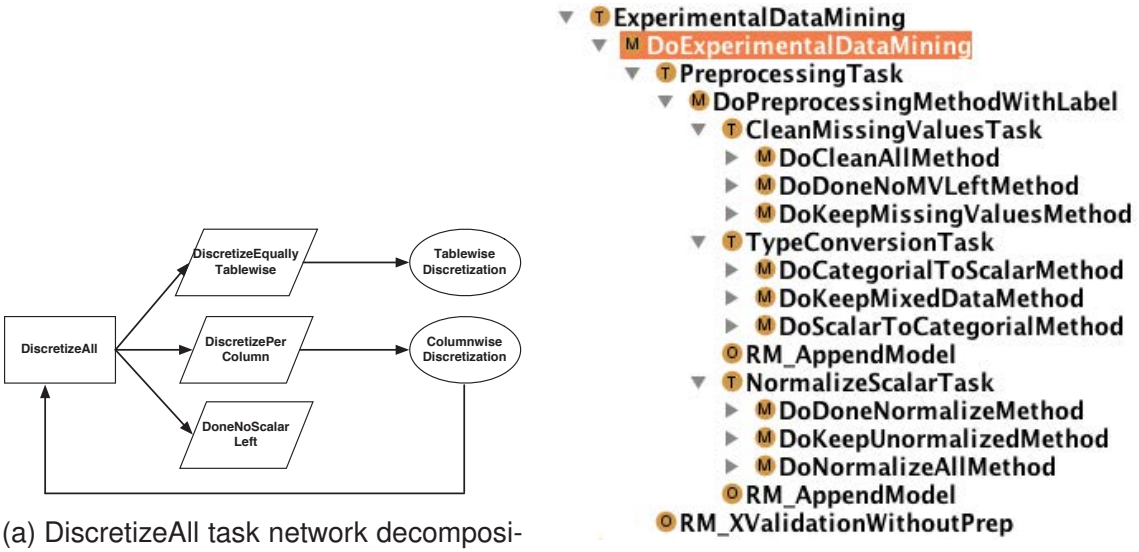
Step 3: *CleanMissingValues*

Step IO: *inputData*=?DINTER, *inputModel*=?PM2, *outputData*=?DOUT, *outputModel*=?PM

Figure 3.7.: CleanMissingValues task with its steps

3.7 illustrates how a complex task is modeled. The *CleanMissingValues* task can be solved by three different methods depending on which condition is satisfied. Each method can define the *IO-Objects* for the task as well as for each of its steps. The *IO-Objects* produced at one step are usually used at the next one so variable bindings are important.

Our HTN grammar contains tasks and operators that work on *DataTable*s directly or on its columns, for example to discretization. To solve this task there are three possible methods (choices): (i) the table is already discretized (*DoneNoScalarLeft*), (ii) all columns are discretized at once (*DiscretizeEquallyTablewise*), and (iii) discretize the columns once by one (*DiscretizePerColumn*). As shown in Fig. 3.8a the last step is a recursive task (since the method *DiscretizePerColumn* can have both tasks and operators as steps) that calls itself until all the columns are discretized.



(a) DiscretizeAll task network decomposition

(b) The main task and its sub-steps

Figure 3.8.: HTN examples

For each step of the KDD process there is a main task that handles its sub-steps. The main task from our ontology is *ExperimentalDataMining* and it contains all the KDD steps as sub-steps as illustrated in Fig. 3.8b.

3.3. An HTN-Planner for KDD Workflows

We treat the problem of designing all the possible KDD workflows for a given dataset and a DM task as an HTN planning problem. In the following we present the components needed for planning (domain, initial and final state) and the planning algorithm used to generate the plans.

3.3.1. The Planning Components

The ontology represents the planning domain, Σ, which consists of operators and other predicates that are used to describe the characteristics of operators, initial state and goal state. Opposed to classical planning, the ontology does not define all the possible states (KDD is a complex process and we are also able to work on attributes therefore this would create a large number of states). For the problem at hand, we need the start and end states and then find a sequence of operators that lead to the goal state. Planning operators correspond to DM operators that are abstract, basic or dominating. Like in classical planning [Nau et al., 2004], operators are triples *o = (name(o), condition(o), effect(o))*, where *name* is an expression of the form $n(x_1, x_2, ..., x_n)$, where n is the name of the operator which needs to be unique and $x_1, ..., x_n$ are the operator's inputs, outputs or parameters. Their conditions and effects are used to decide when to apply a certain operator.

The initial and goal states are defined as a set of ABox axioms (assertional box) consisting of individuals and relations between them. The main task that would solve the DM problem should be selected when defining the main goal. Therefore, the main goal individual has attached a property assertion that stores this information.

HTN-planning allows us to incorporate and exploit the control knowledge from the KDD domain. In addition to actions (**primitive tasks**), HTN planning introduces a set of **methods** that determine how to decompose a task into a set of subtasks or abstract operators. The method descriptions are common **domain knowledge**, which represent **plan fragments**. As defined by [Erol, 1996] there are three types of goals in HTN planning: (i) goal task—desired properties of the final state, (ii) primitive tasks, and (iii) compound tasks– used to decompose the goal tasks. To reduce the complexity of the problem even more we employ **ordered task decomposition** planning, where the tasks are planned in the same order in which they will be executed. The purpose of the planning algorithm is not to achieve a given goal but to decompose the task used to attain the goal. Another improvement that reduces the backtracking of the planning process is to use **planning groups**—find all the primitive tasks (operators) with the same effects and apply them all at once (in this way useless backtracking is avoided).

HTNs can also employ **strategic advice through** *Goals* as proposed in [Myers, 1996]. In our case, advice is expressed via main goals that can have several optional sub-goals. Further restrictions can be defined on the number of subgoals by concept expressions.

3.3.2. The Planning Algorithm

As for classical planning, our planning problem represents a triple $\mathcal{P} = (\Sigma, s_0, g)$, where Σ represents the planning domain, s_0 is the initial state and g a set of goal states. However, by employing HTN planning the problem transforms itself to decomposing the main task (used by the main goal) into a set of primitive tasks (actions or operators), $\mathcal{P} = (\Sigma, s_0, gt)$, where gt represents the goal task network that needs to be decomposed. The solution to the planning problem is the triple (O, s_0, gt), where $O = (o_1, o_2, ..., o_k)$ is a sequence of applicable (basic) operators. Opposed to classical planning, our planning problem translates to finding all the solutions which represents a PLAN GENERATION problem.

Definition 5 PLAN GENERATION *is the following problem: given* $\mathcal{P} = (\Sigma, s_0, gt)$*, find all the possible plans that solve* $\mathcal{P}$*.*

To solve this problem using the algorithm described in Algorithm 1 we need to call the DECOMPOSETASK function with the initial goal task gt[2]. This expands all the applicable methods by decomposing its tasks or expanding its operators. The applicable operators are grouped by their effects and then afterwards applied (e.g., all classification learners produce a model) on the available data. The algorithm will return the set of all possible plans (some steps are plan groups but are expanded and individual plans are returned).

3.4. eProPlan system

eProPlan was developed as a plug-in for the ontology editor Protégé [3] and consists of a set of views that allow modeling the KDD domain, testing operators, generating and visualizing KDD workflows. Protégé 4 offers graphical ontology editing tools with a simple syntax and built-in validation mechanisms. Hence, it provides intelligent assistance for ontology building and also points out the modeling errors through classification, consistency checking and ontology testing. The graphical user interface employs the Manchester-OWL syntax notation [4] to display OWL expressions making them easy to read and efficient to enter. In addition, with the help of a comfortable

[2] This algorithm is not a contribution of this thesis. It is only presented to understand how the workflows are generated.

[3] `http://protege.stanford.edu/`

[4] `http://www.w3.org/TR/owl2-manchester-syntax/`

Algorithm 1 The planning algorithm

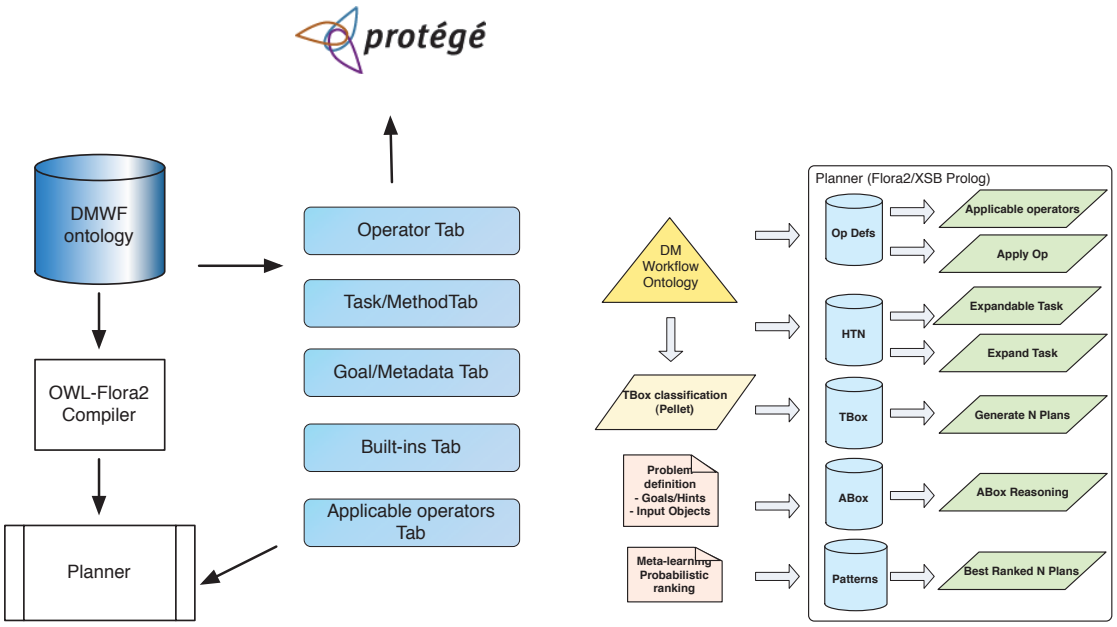
```
function DecomposeTask(t)
    S = ∅
    for all methods m of task t do                    ▷ Go through all methods
        S = S ∪ ExpandMethod(m)
    end for
    return S
end function

function ExpandMethod(m)
    S = ∅
    if condition(m)=true then                         ▷ Method's condition fullfilled
        for all step s of method m do
            if s is a task then
                S = S ∪ DecomposeTask(s)              ▷ Task to be decomposed
            else
                S = S ∪ ExpandOperator(s)             ▷ Operator to be expanded
            end if
        end for
    end if
    return S                                          ▷ List of tasks and operators
end function
function ExpandOperator(o)
    O = ∅
    if condition(o)=true then                         ▷ Operator's condition fullfilled
        if o is abstract then
            O = O ∪ GetApplicableOperators(o)         ▷ Abstract operator
        else
            Apply Operator's effects                  ▷ Applicable operator
            O = O ∪ {o}
        end if
    end if
    return O                                          ▷ List of operators
end function
function GetApplicableOperators(ao)
    O = Get executable operators of ao, group them by effects, apply effects
    return O
end function
```

expression editor the user can rapidly enter OWL class expressions either with mouse or keyboard.

The customized **operator tab** displays for each operator its conditions and effects (both inherited and current). We extended the SWRL editor from Protégé and built our own editor that has a syntax checker and a variable binding checker. The editor supports auto-completion for class, property or individual names as well as for SWRL built-ins. In case of a syntax error, a message is displayed which explains the source of the error and suggests how it can be fixed.

eProPlan allows defining new built-ins which are stored as sub-classes of the *Built-in* concept via the **built-in tab**. Each built-in can have types/parameters and the corresponding implementation in Flora2 [Yang et al., 2003]. Users who want to add new built-ins need to have some Flora-2 knowledge. They have the possibility to define new functions/operations on the data and introduce them in conditions and effects. The built-ins' definition with parameters and implementation is stored as class annotations.



(a) The eProPlan system as a Protégé plug-in

(b) Workflow Ontology and AI Planner capabilities.

Figure 3.9.: eProPlan system and its capabilities

The **task/method tab** allows creating new task/method decompositions via three views. Additionally, it follows the paradigm of advice-taking [Myers, 1996] that enables users to interact with the planning system at high levels of abstraction. The first view

presents a tree with the decomposition of tasks into methods, methods into tasks and operators classes. At the top there are several buttons that can be used for adding, deleting new tasks and methods, creating a new step into the decomposition and moving an existing step up or down in the sequence. The next view displays the conditions and contributions for methods. Each time the user selects a method from the tree on the right side the condition and contribution are shown. We use a similar editor as the one for operators' conditions and effects. The last view displays the method bindings for a selected method. It shows the task to which a method belongs to as head, all the properties related to that task and each step of the method decomposition. For each step in the decomposition we display the properties related to it with default variables that can be edited or removed.

The **goal tab** is used to provide *grounded advice*. Two views allow to define new main goals and their optional subgoals as well as their *IOObjects*.

eProPlan provides support also for plan visualisation via the **applicable operators tab**. It shows the operators that have satisfiable conditions and therefore can be applied for planning. We built a view that shows the applicable operators per individual (input data) allowing the user to choose which operator to apply on the selected parameter. This represents a pseudo-plan stepper since it enables the user to plan step by step by applying operators. The Plan button displays the list of available tasks (individuals) that are assigned to a goal individual. All actions have progress bars that show the progress of the current action and can cancel its execution at any point. The second view is used for plan visualization. Each node is an individual with a label and an icon and represents either an operator or an *IO-Object*. Edges have labels and consist of properties that connect individuals. The modeling of operators is a costly process that requires understanding how the operators work and also how to describe this in the SWRL-like extended language. For this reason, we have a separate tab that allows the developer to check the correctness of the defined conditions and effects only on a selected operator.

The planner is implemented in Flora2/XSB [Yang et al., 2003] and uses different parts from the workflow ontology for different purposes (see Figure 3.9a). Specifically, it employs both a specialized ABox (assertional box—individual assertions) reasoner that relies on an external TBox (terminological box: classes and properties) reasoner (e.g. Pellet[5] or FaCT++[6]) as a subroutine. Since the TBox reasoner can only partially handle OWL2 (Web Ontology Language[7]), we filter all expressions that are not

[5] http://clarkparsia.com/pellet/

[6] http://owl.man.ac.uk/factplusplus/

[7] http://www.w3.org/TR/owl2-profiles/

supported from the ontology. The resulting inferred/completed TBox and its possibly inconsistent class definitions are passed to our ABox reasoner. The ABox reasoner, implemented in Flora2/XSB, first compiles the classified TBox obtained from Pellet on the initial ontology. Then, we process the operators together with their inputs, outputs, preconditions, and effects that are stored as OWL annotations. Tasks and methods are handled analogously. Finally, we finish with the compilation of the problem definition, which is represented by a set of individuals. The problem description has two elements: the input description in terms of meta-data (characteristics of the data like attributes, types, median, etc.) and the goals/hints entered by the user. Both are stored as a set of ABox assertions.

3.5. Evaluation

The two main components of eProPlan are the module that generates the planning domain and the planner. Building the DM ontology is a long-term process that can be always improved and extended. Therefore, it is difficult to measure this process. However, it is interesting to see how fast the system performs on the current DM ontology. KDD is a complex domain and therefore the ontology includes more than 100 concepts and relationships. For this evaluation we use time as a measure for the generation of the domain and the planning process. However, when used on specific datasets the generation time depends on the characteristics of the input data. Therefore, we test it on several datasets that have various data types (this means different attribute groups). In the following we present our findings.

3.5.1. Planning domain generation

eProPlan is a modeling tool that helps to describe and test the KDD domain for planning. Users need to have access to an acceptable fast tool that allows checking the correctness of the domain at any time. For this reason it is important to ensure that the alpha-beta testing does not take too long. Generating the KDD domain as a planning domain consists of the following steps: classify the ontology using and existing TBOX reasoner, compile the TBOX, ABOX, SWRL-rules, normal annotations, annotations and entities for the IDA and ontology built-ins, compile the conditions and effects and finally the HTN. The TBOX classification calls the Pellet reasoner and finds missing relations between concepts. It creates a temporary ontology where the new found inferences are stored and then it is compiled into Flora-2. The ABox, SWRL rules, built-ins and data necessary for the IDA (entities, annotations, etc.) are compiled as

Task Type	Time [ms]
TBox classification	128016
TBox to Flora2	304
ABox to Flora2	2
SWRL to Flora2	26
IDA to Flora2	114
Ontology-builtins to Flora2	2
Load KB in Flora2	10864
Getting inferences	1333
Conditions-effects to Flora2	2160
HTN to Flora2	232
Total classification	140661
Total domain compilation	152858

Table 3.3.: Time needed for compiling each part of the ontology

well. Then the Flora2 files are loaded and new inferences are computed and returned. This is actually a modified version of an TBOX reasoner. Then, if asked the conditions and effects as well as the HTN are compiled.

We have made an evaluation of the various modules in eProPlan. The idea was to measure the time needed to compile different ontologies. This evaluation was done on a MacBook Pro with the following specification: Intel Core 2 Duo, 2.4 GHz, 8 GB RAM. These experiments used the DMWF-HTN ontology from November 2012. The evaluation studies how the system performs on different datasets. It also shows that the system works well for datasets with different sizes no matter how many attributes they have since column groups reduce them to only a few. As we see from Table 3.3 the compilation time to Flora2 is small, but the TBOX classification and the loading of the data into Flora2 as well as the retrieval and insertion of new inferences into the ontology are quite expensive. This evaluation was done only on the domain ontology. In the following we analyze how the data impacts the numbers in particularly the ABOX part as well as loading and getting inferences from Flora2. To achieve this we employ some datasets from the UCI repository: Iris, Labor-negotiations, InternetAds and CommunityCrime datasets. The results are presented in Table 3.5. As we can see the InternetAds dataset, which has the highest number of attributes, takes the longest time to compile and retrieve all the inferences. The results use the normal column approach where all attributes are stored. However, this is very slow for datasets with many columns and therefore the column groups approach was in-

Dataset	#Attributes	ABox to Flora2	Load KB	Get inferences	Classification
Iris	5	4	10273	4958	125596
Labor-negotiations	17	27	9734	12913	122952
InternetAds	1558	285	132174	3610851	3895574
CommunityCrime	128	263	15976	603070	786752

Table 3.4.: How the number of attributes in a dataset impacts the domain compilation time in (ms)

Dataset	#Attributes	ABox to Flora2	Load KB	Get inferences	Classification
Iris	2	3	10607	9927	185442
Labor-negotiations	7	5	11005	13898	16663
InternetAds	4	85	15467	11741	184248
CommunityCrime	6	3	10446	12767	194962

Table 3.5.: How column groups impacts the domain compilation time in (ms)

troduced. To compare the two approaches we have also measured the time for the column group approach. The column group approach has significantly better performance than the normal one especially for datasets which have large amounts of attributes (CommunityCrime and InternetAds). This is due to the reduced number of inferences–less individuals and properties. We are also interested in testing the applicability of operators, the generation and retrieval of plans, etc.

3.6. IDA-API

We developed the Application Programming Interface (API) to automatically generate workflows in current KDD tools. The API provides a set of methods that allow to specify the characteristics of a given dataset, the main task and goal to be solved. Then it can ask for a number of plans and can also choose to have them ranked (either using the default operators frequencies or the approach from [Hilario et al., 2011]). As the ontology models mainly RM operators, we have created a RM plug-in. This is publicly available and it has an installer that downloads and installs all the needed libraries (XSB, Flora2, etc.). An additional plug-in was developed for Taverna since it is able to call RM operators and therefore execute workflows. These two plug-ins prove that the API can be used to get applicable workflows for a certain problem. To be able to use it for other tools (e.g., KNIME, WEKA, etc.) one needs to model all

Method	Description
startPlanner	Starts the connection to the Flora2 planner.
cancelPlanner	Cancels the current running operation on the planner.
shutDownPlanner	Shuts down the connection to the planner.
getMainGoals	Returns a tree with the main goals in the ontology.
getOptionalGoals	Returns a tree with optional goals for a specific goal.
getTasks	Returns all the tasks from the ontology.
getOperators	Returns a tree with all the operators in the ontology.
getPlans	Returns all the possible plans from planner.
refinePlan	Not usable
checkPlan	Checks the correctness of a plan.
createPlanFactory	Allows creating plans for the previous method.
createEmptyGoalSpecification	Allows creating goals, optional goals and meta-data to define the initial problem.

Table 3.6.: Main methods from the IDA-API

their operators similarly to the RM operators.

The IDA-API has the following functionalities: planner operations (*startPlanner, cancelPlanner, shutDownPlanner*), operations on tasks, methods and operators (*getMainGoals, getOptionalGoals, getTasks, getOperators*), operations on plans (*getPlans, refinePlan, checkPlan, loadCase, getCaseIndex*), operations on operators (*createPlanFactory* - allows to add partial plans), and operations on meta-data (*createEmptyGoalSpecification* - allows to define the meta-data and add main and optional goals). The methods are described in more detail in Table 3.6.

3.6.1. KDD in 7 clicks

The RapidMiner plug-in that relies on the IDA-API allows to completely automate and simplify the KDD workflow design. Without the RM-IDA data mining is typically achievable by highly-trained professionals such as DM consultants. They have to know a lot about DM methods including details about their implementation in specific KDD tools. They have to inspect the data and combine the operators into an adequate workflow.

The IDA reduces the technical burden, it now offers ”DM with 7 clicks” (see Figure 3.10). (1) Show the IDA-Perspective of the tool; (2) drag the data to be analyzed from the repository to the view or import (and annotate) your data; (3) select your main goal in DM; (4) ask the IDA to generate workflows for data and goal; (5) evaluate all plans

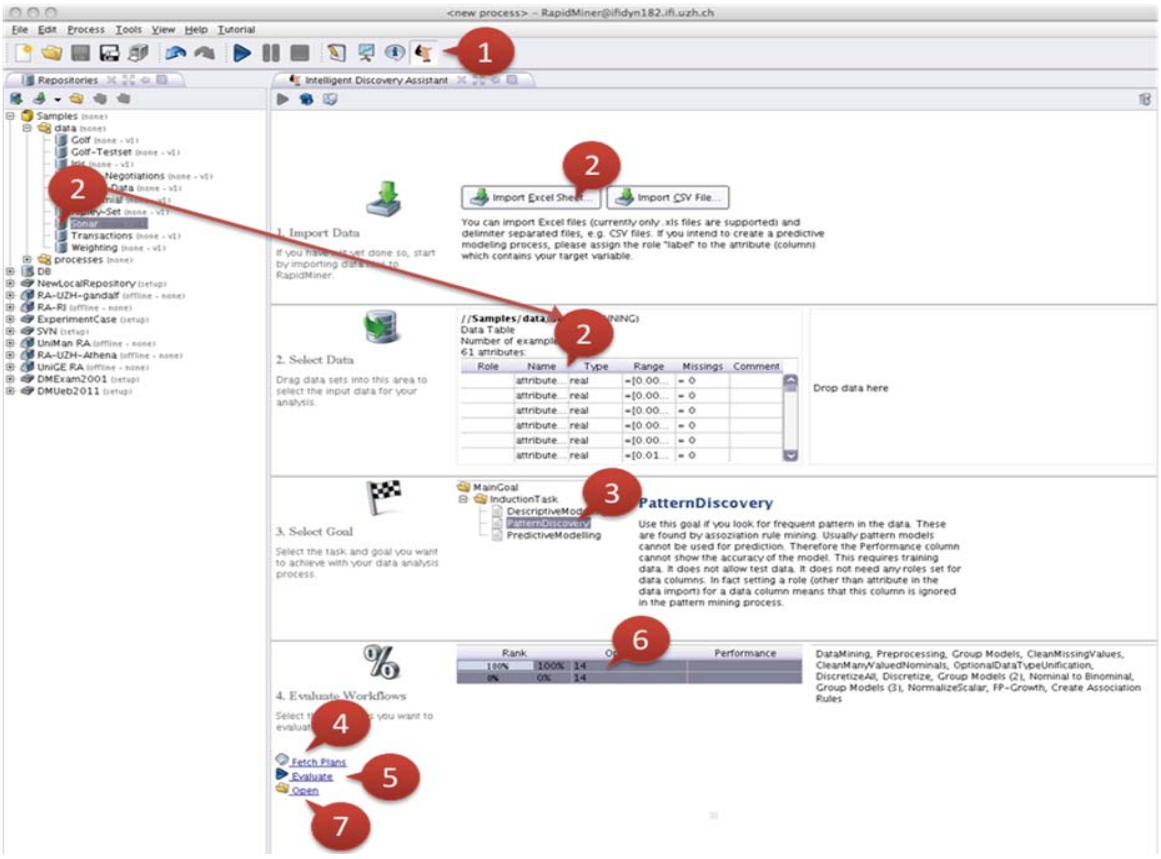


Figure 3.10.: IDA Interface in RapidMiner

by executing them in RapidMiner; (6) select the plan you like most to see a summary of the plan (the screenshot in Figure 6 is made after this step); and finally, (7) inspect the plan and its results. Note that these steps do not require detailed technical knowledge anymore. Still a user should be aware of what (s)he is doing when (s)he uses DM, i.e. (s)he should know the statistical assumptions underlying DM (e.g., a user should know what it means to have a sample that is representative, relevant, and large enough to solve a problem with DM/statistics). But this is knowledge required in any experimental science.

To improve the user experience with the RM-IDA plug-in we have developed a simple installer based on precompiled binaries. It works on Linux, Mac OS X 10.5/6, Windows 7 and Windows XP systems. The RapidMiner IDA Extension can be downloaded (or even auto-installed) from the Rapid-I Marketplace [8].

3.7. Discussion

In this section we have introduced a sustainable method for automating the KDD process as follows: First, we have used an ontology that was modeled based on DM experts' knowledge and on DM cookbooks (CRISP-DM standard). The ontology has a set of features inspired from previous work in the field and from AI planning that aligns it to the planning domain (conditions, effects, task and operator hierarchy, etc.). Second, we have developed a set of tools that allows maintaining, extending and testing the ontology over time. This confirms the hypotheses from RQ3. eProPlan does not only generate correct plans, but also filters the useless ones by allowing to define tasks and methods. eProPlan improves the expert's experience with modeling and describing the KDD domain by providing a set of tabs specialized to achieve DM tasks. Modeling and testing are made easier since they are integrated together in Protégé. Also the IDA-API allows to plug-in the planner capabilities into KDD tools to generate workflows automatically starting from a the dataset's characteristics and the task description. This is the first IDA that provides such capabilities.

[8] http://rapidupdate.de:8180/UpdateServer/faces/product_details.xhtml?productId=rmx_ida

4

Ranking KDD Workflows

A system that automatically generates workflows does not simplify the data analysis process for users. The eProPlan/eIDA system can generate thousands of workflows for a specific problem and dataset [1]. Therefore, there is a need to filter the number of workflows returned to the user by ranking them and recommending only a limited number of workflows (e.g., TOP N workflows). In this section, we explore how collaborative filtering and auto-experimentation can be combined to achieve this goal.

4.1. Introduction

Intelligent Discovery Assistants (IDAs) [Bernstein et al., 2005, Serban et al., 2012] employ planning techniques to automatically generate KDD workflows. Many of them only enumerate all the possible correct combinations of operators for each KDD step. However, the choice of a good performing workflow is left to the user. Despite the existing systems and research in the field, the question of how to select among the resulting workflows has not yet been adequately addressed. Whilst some favor heuristics [Bernstein et al., 2005] or meta-learning [Hilario et al., 2011] none of those provide a satisfactory solution. For this purpose, we propose to combine

[1] The number of workflows depends on the operators modeled in the ontology.

auto-experimentation (also suggested by [Bernstein et al., 2005]) with collaborative filtering (CF) to systematically explore the space of proposed workflows to obtain a good ranking. Specifically, we use eProPlan/eIDA [Kietz et al., 2010a] to generate all the workflows for several datasets and create a case-base containing execution details for each workflow. When confronted with a new dataset our system explores the design space of possible workflows using the same IDA, then chooses to run a subset of experiments, and finally employs the case-base to suggest which workflows are likely to be the best performing. Hence, our approach combines Ontology-based planning, auto-experimentation, and CF to recommend good KDD workflows – a goal from which even DM experts can profit, as they usually use only a small subset of the available operators [Kohavi et al., 2000].

The core idea of our approach lies in the observation that composing good KDD workflows should follow Herb Simon's model of problem solving [Newell et al., 1958]:

a) Problem definition

b) Design of alternatives

c) Choice of the best alternatives

In KDD problems, the *problem definition* is given by the data set to be explored and the desired analysis task with the associated user-desiderata [Bernstein et al., 2005]. In our case the user provides the dataset and task description. Then the characteristics of the data are extracted and used for planning.

The *design of alternatives* step is essentially a planning problem, where the design-space of possible KDD workflows is explored. As introduced in Section 2.3 and in detail in the work of [Serban et al., 2012] a number of KDD workflow planning systems exist. For our experiments we employ eProPlan [Kietz et al., 2010a], which works on a large number of operators. Hence, a practical system would take a user's problem definition and pass it to the planner for the enumeration of the KDD workflow design space. The result is a large list of applicable KDD workflows.

One of the main contributions of this thesis is a robust and well-performing method for the *choice of the best alternatives* based on auto-experimentation and CF. Specifically, we select a subset of workflows from the planner's output, employ them for auto-experimentation, and then use the results of the experiments as an input to a CF inspired method that suggests how to rank all the applicable workflows.

First, we introduce CF as it is used in general and then describe how we adapted it to solve the problem of ranking KDD workflows. Second, we tackle in detail the

cold-start problem, meaning when confronted with a new dataset what workflows need to be executed such that CF can be applied. Third, we analyse the impact of sparsity in the CF matrix, when datasets have only a certain number (relatively small) of executed workflows.

4.2. Collaborative Filtering

The virtual world of documents or images on the Web would be useless without a recommendation engine that retrieves the products most likely to be ”interesting” for a user as suggested by the Long Tail claim [Anderson, 2010]. The difference between search engines and recommendation systems is that the latter do not query the results a specific user wants to find. Instead the recommendation relies on the user preferences (history) and the community data (other users' preferences).

Nowadays many companies use CF to recommend products to their users. As more and more online platforms are available where users can have their own profile, CF has become very important for e-commerce applications [Schafer et al., 2001]. The main contributions of recommender systems for such applications are: (a) help customers to find products they wish to purchase, (b) improve cross-sell by suggesting additional products, and (c) gain customer's loyalty by learning what he/she likes and building a very accurate profile—provide customized interfaces to each user.

For the sake of self-containedness, this section provides a brief introduction to collaborative filtering, referring the interested reader to [Melville and Sindhwani, 2010] for a comprehensive presentation.

Recommender systems are categorized in two broad types: *content-based* and *collaborative filtering* approaches. Content-based approaches basically rely on a feature-based description of the considered options, and exploit the user's history to determine the options most similar to the previous options positively rated by the user. CF approaches instead leverage the usage data [Bell et al., 2009] by recording the ratings of many users for various items. Informally, CF aims at uncovering the similarity among the options, and among the users, hidden in the usage data. These similarities are further used to recommend options similar to those the user liked, *or options liked by similar users*. Naturally, both approaches can be combined to get the best of both worlds [Su and Khoshgoftaar, 2009]). CF involves memory-based or model-based approaches. In memory-based approaches, an explicit similarity among users is extracted from their past ratings of the items in the application, using e.g. Pearson correlation coefficient or the cosine similarity (Table 4.1). The drawback of

memory-based approaches is that they require sufficiently many users to have rated many items for the similarity to deliver reliable indications.

In model-based approaches, a model of each user (or each item) is built based on the community data. Most common approaches employ user clustering and bayesian models. Other approaches include Latent Semantic Indexing and probabilistic Latent Semantic Indexing [Hofmann, 1999], Latent Dirichlet Allocation [Blei et al., 2003], Matrix Factorization [Rennie and Srebro, 2005], and Ranking-based approaches [Weimer et al., 2007].

Both memory- and model-based face the cold-start issue [Schein et al., 2002]: they offer little insight into the preferences of a brand new user, and active learning approaches are used to efficiently explore the option space [Harpale and Yang, 2008, Boutilier et al., 2010].

Next, we introduce the most used CF methods focusing on their advantages and disadvantages. We also analyze the cold-start problem and possible solutions.

4.2.1. Collaborative Filtering Methods

For a clear understanding of the domain we shortly present some of the most important methods in the CF field. Most common approaches from CF are *similarity-based methods*, *matrix-decomposition methods* and *ranking-based methods*. The collaborative filtering problem can be defined as: given a matrix $\mathcal{R}$ of ratings, where rows are users and columns are items, the value in the matrix represents a rating given by user i to item j. For new users we may have none or only a few rated items. The problem that CF tries to solve is to recommend new users based on their rating history and their similarity with other users/items.

Similarity-based methods Existing approaches use various similarity metrics [Breese et al., 1998] to find similar users based on their ratings. These are user-based approaches that allow to find the affinity of the user for non-rated items. Another idea uses the similarity between items to compute the ratings [Sarwar et al., 2001]. The disadvantage of these methods is the fact that their performance decreases with the sparsity of the matrix [Breese et al., 1998]. And in most applications the rating matrix is quite sparse.

Matrix-decomposition methods The Netflix competition and recent results in recommender systems have shown that matrix-factorization techniques give more accurate recommendations than memory-based methods [Koren et al., 2009] since addi-

tional information can be incorporated to improve the prediction model. This approach maps users and items to a joint latent factor space where user-item interactions are modeled as inner products in that space. The model is closely related to SVD [Ma, 2008], however SVD is applicable only on a full matrix. To counteract this problem recent methods model directly the observed ratings through a regularized model. Different learning algorithms [Koren et al., 2009] are used to minimize the regularized square error on the set of known ratings and learn the factor vectors. This allows to find the missing ratings and obtain a full matrix.

Ranking-based methods Previous methods predict the missing ratings from the existing ones, however there are applications that only need the Top N ratings (Web shops to recommend new items). The ranking methods make a structured prediction: they predict the relative order of the ratings instead of the absolute value. One of such approaches is CofiRank [Weimer et al., 2007] which extends the maximum-margin matrix factorization approach [Srebro et al., 2004]. The obtained ranking scores are good and the method scales well on CF tasks.

4.2.2. Cold-start problem

Formally, CF aims at recommending new items to users, relying on the previous items they liked (user history). It exploits a (user $\times$ item) matrix $\mathcal{R}$ describing which items have been liked/disliked by which users (community data). Indeed, matrix $\mathcal{R}$ is overwhelmingly void, in the sense that an average user has seen a handful of items overall (e.g. 1% in the famous Netflix problem [Bell and Koren, 2007], .01 in the Yelp rating problem [Fennell, 2009] or the million-song dataset challenge [McFee et al., 2012]). The problem arises when for a new user the matrix has no ratings or only a few, or similarly for new items.

Existing research combines both CF and content-based methods to solve this problem [Schein et al., 2002, Lam et al., 2008]. These hybrid approaches are then able to produce ratings for new items or users based on other characteristics of users/items. Additional information about the user can be gained by interacting with the user, e.g. using an *active learning* approach [Rubens et al., 2011, Harpale and Yang, 2008]. Here the user can actively help in the selection of the target items for which she wants to gets the ratings.

Next, we shortly introduce some of the most successful applications areas of CF.

4.2.3. Systems Using Collaborative Filtering

As development in technology supports e-commerce and other kind of e-businesses, more users join and use these platforms. In addition, social networks have become very attractive nowadays being a trend, a style of life or a 'must do' (e.g., Facebook, LinkedIn, etc.). Thus, providing good recommendations to people is essential for improving their services. Applications relying on CF range from information (e.g., Google News,) to entertainment (e.g., Netflix, IMDb), e-commerce (e.g., Amazon, E-bay) and social networks (e.g., Cyworld, Twitter, Facebook, etc.).

Google news The Google approach uses CF to generate news recommendations for users based on their click history and on more scalable algorithms than the memory-based ones [Das et al., 2007]. As such several model-based algorithms were tested and tailored to their needs: MinHash - probabilistic clustering method for user clustering, LSH - use hash functions to determine near neighbors, PLSI - probabilistic latent models which finds user communities and item communities. To achieve better scalability the algorithms were adapted and implemented using the Map-Reduce paradigm [Dean and Ghemawat, 2008].

Netflix and Movies In 2006 Netflix [2] has initiated a contest for finding the best CF algorithm to predict movie ratings. It released a large database of 100 million movies and TV shows ratings from around 500000 users. This contest contributed to the spread of CF in several domains and lead to the development of new interesting and well-performing CF algorithms [Bell and Koren, 2007, Koren, 2009].

E-commerce This area can profit from CF since it has information about the preferences of users regarding products.

Amazon.com uses recommendations extensively mainly as a marketing tool in email (Eyes or Amazon.com Delivers) or web sites [Linden et al., 2003]. This can be easily seen on the Amazon.com website. Here, users can refine their recommendations on different categories of products (Your recommendations, Customer comments), rate their purchases, etc. The secret to their recommendations is an item-item CF algorithm that instead of finding similar users it finds similar products or items.

E-bay allows both sellers and buyers to provide satisfaction ratings for the people they have done business with [Schafer et al., 1999]. Customers can browse the

[2] http://www.netflixprize.com/

sellers profiles and decide based on their ratings if they want to do business with them or not. Another interesting feature in E-Bay is the *Personal Shopper* that allows users to define filters for items they are interested in [Schafer et al., 2001]. For example, one could define a period limit and select a set of keywords to show the interest areas. Then the customer receives regularly notifications about the results of the search.

Social networks Social networks gather information about how people connect to each other in groups of friends with similar interests (like Facebook, Google+, Cyworld, Myspace, etc.).

Cyworld [Liu and Lee, 2010] use Cyworld data to show how traditional CF methods can be improved by including social network information into CF. Here, people with similar ratings on items are also differentiated by the social groups they are part of (strange people can have similar taste but friends are closer to you). The results indicate that this approach is more accurate for social networks than classic CF. Similar approaches can probably be used in most of social networks.

Twitter can be used to create links and connections between users. [Hannon et al., 2010] developed Twittomender a system that is able to recommend whom to follow on Twitter based on the user's current profile and his history (recent tweets, etc.). The system automatically extends its database each time a user allows it to connect and use information from his/her account. It basically combines content-based CF with classic CF.

4.3. Collaborative Filtering For KDD Workflows

To address the task of selecting representative workflows (previously introduced as *choice of best alternatives*) we propose to combine auto-experimentation with collaborative filtering. The core of this idea is illustrated in Fig. 4.1. First, we use a set of datasets to create a case-base of ratings. The ratings reflect the performance of a workflow for a given dataset. Specifically, we execute all possible plans for each of these datasets and enter their performance into the case-base. This can happen long before any given analysis, effectively trading off space versus time, or it can even be build incrementally, adding the performance of the executed experiments for future ranking predictions. Second, when a user analyses a new dataset the system takes the planner's output, chooses a subsample of these workflows for auto-experimentation,[3] and uses the results to run a CF process on the case-base to rank

[3]Tackling the cold-start problem

all the workflows.

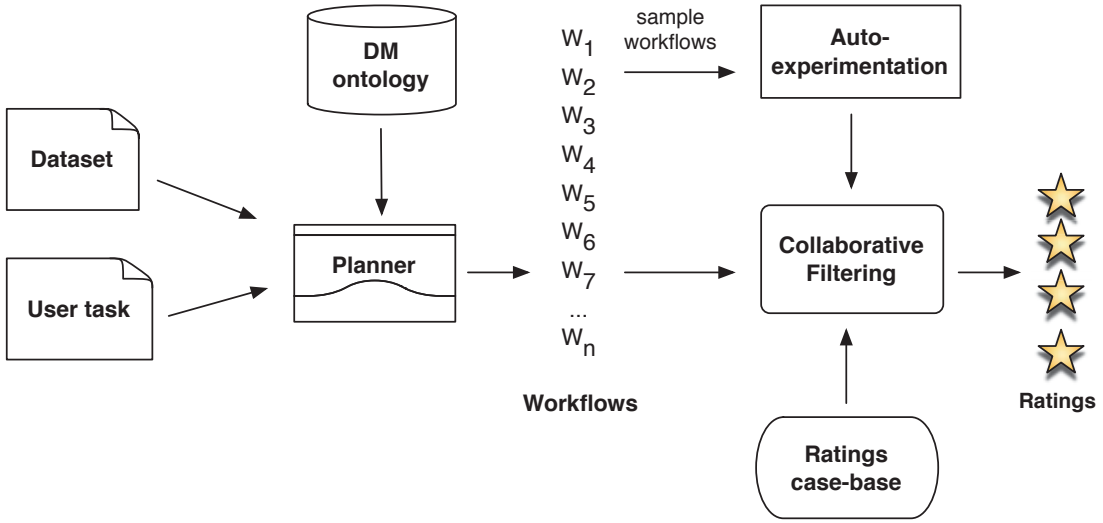


Figure 4.1.: Collaborative-Filtering/Auto-Experimentation Based IDA

In this section first, we shortly introduce auto-experimentation, second we discuss how we transformed traditional CF to rank KDD workflows and third we explore different methods that are used in CF.

4.3.1. Auto-experimentation

All the correct workflows for a given problem and task at hand are generated via eProPlan and eIDA as described in Section 3.4. This represents the design of alternatives—exploration of the entire KDD space.To apply CF to rank workflows we need a case-base of execution results for several datasets. We, therefore, selected datasets from the UCI Irvine [Frank and Asuncion, 2010] and other ML and DM repositories (TunedIT [4]) and executed all the possible workflows. For each workflow we have recorded the duration of execution and the performance of the workflow (accuracy, error, etc.). In our experiments, we focus solely on classification and regression problems [5].

To illustrate the extent of the design space explored by eProPlan we provide first a pseudo-code version for the tasks and methods in Fig. 4.2. In the HTN grammar each pre-processing task has three options: (1) replace all the columns that have a certain property (missing values, not normalised, change a type, etc.), (2) there are

[4] http://tunedit.org

[5] The ontology can be extended such that it generates workflows also for other type of problems.

ExperimentalDataMining := DoPreprocessingMethodWithLabel RM_XValidation

DoPreprocessingMethodWithLabel := CleanMissingValuesTask TypeConversionTask NormalizeScalarTask

CleanMissingValuesTask := DoCleanAllMethod | DoDoneNoMVLeftMethod | DoKeepMissingValuesMethod

TypeConversionTask := DoCategorialToScalarMethod | DoKeepMixedDataMethod | DoScalarToCategorialMethod

NormalizeScalarTask := DoDoneNormalizeMethod | DoKeepUnormalizedMethod | DoNormalizeAllMethod

DoCleanAllMethod := RM_Replace_Missing_Values_all

DoDoneNoMVLeftMethod := RM_Empty_ModelGroup

DoKeepMissingValuesMethod := RM_Empty_ModelGroup

DoCategorialToScalarMethod := CategorialToScalarConversionTask

DoKeepMixedDataMethod := RM_Empty_ModelGroup

DoScalarToCategorialMethod := DiscretizationOutsideValidationTask

DoNormalizeAllMethod := Normalize_all

DoDoneNormalizedMethod := RM_Empty_ModelGroup

DoKeepUnormalizedMethod := RM_Empty_ModelGroup

CategorialToScalarConversionTask := DoCategorialToScalarAllMethod

DiscretizationOutsideValidationTask:= DoDiscretizeTablewiseOutsideMethod

DoCategorialToScalarAllMethod := RM_Nominal_to_Numerical_all

DoDiscretizeTablewiseOutsideMethod := DiscretizeOutsideValidation_all

Figure 4.2.: Task/Method grammar for generating workflows

no column with that type and therefore an empty model is used, and (3) there are columns with that property but keep them as they are (keep the missing values, do not normalise, do not convert the type, etc.). Each method has attached a condition that guides the planner to choose the correct one. We have defined these three methods to see how the workflows perform if the preprocessing is done compared to no preprocessing.

A second pseudo-code simplified version for the basic operators of the HTN planning grammar is shown in Fig. 4.3. For the ease of understanding we have simplified the grammar by removing the concept of tasks/methods to ensure a good flow. The grammar follows the KDD steps (preprocessing, feature selection, data mining and evaluation).

Both representations describe the type of workflows that are generated depending on the characteristics of the dataset.

4.3.2. Memory-based CF

Similarly to recommender systems [Melville and Sindhwani, 2010] our goal is to recommend the most appropriate workflows for new datasets.

Suppose that a case-base stores the execution results of m workflows on n datasets.[6] Let matrix $\mathcal{R} \in \mathbb{R}^{n \times m}$ be the matrix that stores the ratings of all workflows W on every considered dataset D and I_{ij} be the indicator of the presence of a score such that if $I_{ij} = 1$ the dataset $d_i \in D$ has a score for the workflow $w_j \in W$ and 0 if the score is missing. In our case, the rating represents a workflow performance measure (such as accuracy, AUC, or RMSE). By construction, the decomposition of $\mathcal{R} = D\Delta W^t$ achieves the goal of collaborative ranking, i.e. predicting how a known workflow will perform on a known problem. Our goal, however, rather corresponds to the "cold-start" problem [Schein et al., 2002]: we want to know which workflow(s) are most suitable to a (new) dataset at hand (denoted as target dataset d_t). In contrast to traditional collaborative filtering, the planner provides us with the set of applicable/possible workflows ($P \subseteq W$), thus reducing the number of workflows that need to be ranked.

To employ CF, the first step is to sample and execute a set of workflows S (selected workflows $S \subset P$), to provide the initial ratings of the target dataset such that $\forall w_j \in S : I_{d_t,w_j} = 1$. Given the ratings' matrix we can apply classic memory-based CF-methods [Breese et al., 1998] to generate the ratings for the applicable workflows in P.

[6] Note that not all n data sets will have results for m workflows, as some of the workflows may not be applicable.

Workflow := Preprocessing* 10-fold_XVal(Training Evaluation)

Preprocessing := CleanMissingValues TypeConversion Normalization

Training := FeatureSelection* PredictiveSupervisedLearners

Evaluation := ApplyModel EvaluateModel

CleanMissingValues := CleanAll | KeepMissing

TypeConversion := CategorialToScalar | ScalarToCategorial | KeepMixed

Normalization := NormalizeAll | KeepUnormalized

FeatureSelection := AttributeWeighting SelectByWeights

EvaluateModel := PerformanceBinominalClassification | PerformanceClassification
| PerformanceRegression

CleanAll := ReplaceWithMin | ReplaceWithMax | ReplaceWithAvg | ReplaceWith0

CategorialToScalar := NominalToNumericalAll

ScalarToCategorial := DiscretizeAll

DiscretizeAll := ByBinning | ByFrequency | BySize

NormalizeAll := ByZTransformation | ByRangeTransformation

AttributeWeighting := DataToWeights | ByChiSquaredStatistic | ByCorrelation
| ByDeviation | ByGiniIndex | ByPCA | ByRule | BySVM
| ByInformationGain | ByInformationGainRatio | ByRelief
| ByUncertainty | ByValueAverage | ByUserSpecification

PredictiveSupervisedLearners := AutoMLP | CHAID | DecisionStump
| FastLargeMargin |ID3| DecisionTree | LinearDiscriminatAnalysis
| LogisticRegression | LogisticRegressionEvolutionary | NaiveBayes
| NaiveBayesKernel | Perceptron | QuadraticDiscriminantAnalysis
| RandomForest | RandomTree | RegularizedDiscriminantAnalysis
| RuleInduction | SingleRuleInduction | SingleRuleInductionSingleAttribute
| SubgroupDiscovery | SVMPSO | LibSVM-MCSVC | LibSVM-nuSVC
| VectorLinearRegression | LocalPolynomialRegression| LinearRegression
| UnrelatedRegression | GaussianProcess | LibSVM-epsilonSVR
| LibSVM-nuSVR | NeuralNet | relevanceVectorMachine | SVM
| SVMEvolutionary | SVMLinear | kNNBregmanDivergences
| kNNMixedMeasures | kNNNominalMeasures | kNNNumericalMeasures
| HyperHyper | DefaultModel

Figure 4.3.: Basic operator grammar for generating workflows

In this thesis, we use the *nearest-neighbor*–kNN, which generates ratings for the target dataset based on the ratings of the dataset's/workflow's best neighbours. The *dataset-based CF* predicts the ratings for P on the existing ratings in S and the similarity to other datasets as follows:

$$\widehat{\mathcal{R}}_{d_t,w_j} = \overline{r}_{d_t} + \alpha \sum_{i=1}^{n} sim(d_t, d_i) \cdot (r_{d_i,w_j} - \overline{r}_{d_i}) \tag{4.1}$$

where $\widehat{\mathcal{R}}_{d_t,w_j}$ is the rating for the target dataset d_t and workflow w_j and $\overline{r}_{d_t}$ is the mean rating for the target dataset,

$$\overline{r}_{d_t} = \frac{1}{|S|} \sum_{i \in S} r_{d_t,w_i}$$

and w_j are the workflows of the target dataset that need to be rated. The weights $sim(d_t, d_i)$ can reflect distance, correlation or similarity between each dataset d_i and the target dataset. α is a normalising factor such that the absolute value of the weights sums to unity. Common measures for similarity between datasets are *Pearson correlation*, *cosine similarity*, or *PIP* as illustrated in Table 4.1. We use PIP as it has been shown to better handle cold-start problems [Ahn, 2008].

In contrast to the dataset-based CF approach, *workflow-based CF* [Sarwar et al., 2001] compares the target dataset's already rated workflows to the ones in the case-base to predict the unknown ratings. Specifically, the prediction is computed as a weighted sum of the most similar workflows as follows:

$$\widehat{\mathcal{R}}_{d_t,w_j} = \frac{\sum_{i=1}^{K} sim(w_j, w_i) r_{d_t,w_i}}{\sum_{i=1}^{K} |sim(w_j, w_i)|} \tag{4.2}$$

where the w_j are the workflows for which we need a rank and w_i the workflows from the selected set S. This approach captures the performance of the target dataset for similar workflows. Here, the similarity of two workflows can be computed using cosine similarity, Pearson correlation, or adjusted cosine similarity that offsets the differences in the rating scale between different datasets (cf. Table 4.1). To improve the predictions a regression model can be used to predict the values for the similar items as follows:

$$\bar{R}'_N = \alpha \bar{R}_i + \beta + e$$

where N is the similar item and i is the target item [Sarwar et al., 2001].

Slope One An improved version of the item-item algorithm is slope one [Lemire and Maclachlan, 2005]. It reduces overfitting by using a single parameter in the linear

Measure	Definition
Pearson's correlation	$sim(d_t, d_i) = \frac{\sum_j (r_{d_t,w_j} - \overline{r}_{d_t})(r_{d_i,w_j} - \overline{r}_{d_i})}{\sqrt{\sum_j (r_{d_t,w_j} - \overline{r}_{d_t})^2 \sum_j (r_{d_i,w_j} - \overline{r}_{d_i})^2}}$ where w_j represents the workflows that were executed for both dataset d_t and d_i. The datasets' means are computed only for ratings on co-rated workflows.
Cosine	$sim(d_t, d_i) = \sum_j \frac{r_{d_t,w_j}}{\sqrt{\sum_{j \in I_{d_t}} r^2_{d_t,w_j}}} \frac{r_{d_i,w_j}}{\sqrt{\sum_{j \in I_{d_i}} r^2_{d_i,w_j}}}$ w_j represents the common workflows of both d_t and d_i.
Adjusted cosine for workflows	$sim(w_i, w_j) = \frac{\sum_u (r_{u,w_i} - \overline{r}_u)(r_{u,w_j} - \overline{r}_u)}{\sqrt{\sum_u (r_{u,w_i} - \overline{r}_u)^2 \sum_u (r_{u,w_j} - \overline{r}_u)^2}}$ offsets the differences in the rating scale between different datasets
PIP	$sim(d_t, d_i) = \sum_j PIP(r_{d_t,w_j}, r_{d_i,w_j})$ where w_j belongs to the set of co-rated workflows by both d_t and d_i. $PIP(r_1, r_2) = Proximity(r_1, r_2) \times Impact(r_1, r_2) \times Popularity(r_1, r_2)$, where Proximity, Impact and Popularity are defined in [Ahn, 2008]

Table 4.1.: Common similarity measures in CF

regression ($f(x) = x + b$, where b is the free parameter). For some applications slope one performs better than other CF algorithms. We include it for comparison with the item-based approach.

Summarising, we propose to employ CF to produce recommendations based on a pre-computed case-base in the form of the dataset$\times$workflow matrix $\mathcal{R}$. Dataset-based CF and workflow-based CF differ in how they employ $\mathcal{R}$.

4.3.3. Model-Based CF

The previously introduced methods are memory-based CF and employ k-NN with similarity measures tailored to the structure of the data (sparsity). We have also explored methods that build more sophisticated models, like clustering and SVD. In the following we shortly present some of the tested methods.

4.3.3.1. Clustering

Several clustering techniques are used in CF. The first method, clustering on the items, is used to alleviate the sparsity of the matrix. The second one clusters the users, to find similar groups of users. The first method is more a form of pre-processing than a prediction method [OConnor and Herlocker, 1999]. Unlike item clustering, user clustering can be used to predict the missing ratings for the active users. In this thesis we have employed the k-medians clustering described in [Marlin, 2004] as it has been shown to perform well. This is a modified version of the k-means algorithm, that replaces the squared distance as an objective function with the absolute distance. Here, the optimal centroids are computed as the median of the points in the cluster. The algorithm uses an iterative optimisation procedure that tries to find the local minimum of the objective function. As opposed to the classic k-means the missing values are not considered, but they are excluded from the computation. The centroids for the initial clusters are chosen randomly. The algorithm finishes when the function converges, meaning that the distance function does not change anymore as well as the assignment of points into clusters. In our case we have treated the non-applicable workflows as missing values. The algorithm takes into account the number of ratings to avoid selecting datasets that have few rated workflows in common with the target dataset. Therefore, when re-computing the cluster for a dataset we select the one for which it has the minimum distance but also a high number of common rated workflows (at least half from the possible ones) as suggested by the author. The modified

version of the algorithm used in our experiments is presented in Algorithm 2.

Algorithm 2 The KMedians clustering with missing values algorithm

function KMEDIANS-LEARN(r, K)
  Initialize p_k to k random datasets
  while $F(r, c, p)$ not converged **do**
    for $u = 1$ to N **do**
      $c_u \leftarrow \text{argmin}_k\{\sum_{\{y|r_{u,y},p_{k,y}\neq\perp\}} |r_{u,y} - p_{k,y}|\}$
    end for
    for $k = 1$ to K, $y = 1$ to M **do**
      $p_{k,y} \leftarrow \text{median}\{r_{u,y}|c_u = k, r_{u,y} \neq\perp\}$
    end for
  end while
  return p
end function
function KMEDIANS-PREDICT(r_a, p, K)
  $l \leftarrow \text{argmin}_k\{\sum_{\{y|r_{a,y},p_{k,y}\neq\perp\}} |r_{a,y} - p_{k,y}|\}$
  $\hat{r}_{a,y} \leftarrow p_l$
  return p_l
end function

4.3.3.2. Singular Value Decomposition

SVD is a matrix factorization method that reduces the dimensionality of data by mapping a high dimensional input space into a lower dimensional latent space. A matrix is therefore reduced to several low rank matrices. Given a matrix R of size $N \times M$, the singular value decomposition of R is a factorization

$$R = U\Sigma V^T$$

where U has size $N \times M$, Σ has size $M \times M$, and V size $M \times M$. Σ is a diagonal matrix, both U and V are orthogonal matrices where U contains the left singular vectors and V^T the right singular vectors. In our case, the matrix R contains the ratings given by the datasets to each workflow. U is the dataset-to-groups affinity matrix, V is the workflows-to-groups matrix, and the diagonal elements of D represent the 'expressiveness' of each group in the data (groups could be for example types of workflows). Having these two matrices one can predict the missing scores by $p(U_i, M_j) = U_i^T V_j$.

Classic SVD can only be applied on matrices with no missing values. To show the performance of SVD approaches we have devised two use cases. First, we use SVD for reducing the dimension of the datasets features, then we tested a proposed version of SVD that learns the U and V vectors incrementally. This allows to use SVD also for sparse matrices.

SVD for Clustering First we employ SVD to find clusters of datasets. SVD finds affinities for datasets to certain groups and the resulted matrix U can then be used to cluster them. The missing values in the matrix R represent non-applicable workflows and for the target dataset also missing ratings. We apply SVD on the matrix and find the affinities of both datasets and workflows for different groups in matrices U respectively V. As the rating matrix has missing values and classic SVD does not work well under these conditions, we employ a special variant of SVD for sparse data ($svds$ from Matlab). We devised the algorithm presented in Algorithm 4. First, the clusters of datasets are computed based on the U matrix. Then, for each new target dataset we select the most popular workflows and execute them. Given a set of ratings for the target dataset, we compute the similarity to each of the centroids (the centroids are surrogate users which contain the average rating for each item in the cluster). The ratings of the target dataset are then computed using the l most similar clusters as follows

$$\widehat{\mathcal{R}}_{d_t,w_i} = \overline{r}_{d_t} + \alpha \sum_{i=1}^{l} sim(d_t, c_i) \cdot (r_{c_i,w_j} - \overline{r}_{c_i}) \tag{4.3}$$

where r_{c_i,w_j} is the rating of the centroid of cluster c_i and $\overline{r}_{c_i}$ it is the mean of the ratings in the cluster. As a similarity we used cosine similarity and the α factor represents the weighting factor (1 divided by the sum of all weights or similarities for the selected clusters).

Incremental SVD The sparsity in the CF matrix does not allow to apply SVD directly. To solve this issue people have devised several methods of learning the two low rank matrices, U and M. This idea proved to be successful in the Netflix prize [Bell and Koren, 2007, Koren et al., 2009, Koren, 2009] and even outperform the kNN method. We have followed closely the methods presented in [Ma, 2008] and selected the Complete Incremental Learning SVD version as it proves to give good results. The idea behind the algorithm is to find optimal U and M that minimize the sum of squared errors between the ratings in the matrix and the new predicted values, where

Algorithm 3 SVD-based clustering

- Apply SVD on the matrix $R = UDV^T$
- Cluster the datasets using the affinity matrix U in K clusters
- Order the clusters descending by their size

while $numWfs < numSel$ **do**

  for $i = 1$ to K **do**

    - Select most popular workflow

  end for

end while

- Compute similarity to all clusters
- Select l most similar clusters
- Compute ratings

the function is

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(R_{ij} - p(U_i, V_j))^2 + \frac{k_d}{2}\sum_{i=1}^{n} |||U_i||| + \frac{k_w}{2}\sum_{i=1}^{m} |||V_j|||,$$

where k_d and k_w are regularization coefficients for datasets respectively workflows to prevent overfitting. Ratings in CF are usually bound to an interval $[a, b]$(e.g., $[1-5]$) and therefore the prediction function should also be bound to values in $[0, b-a]$. The prediction function would therefore become

$$p(U_i, V_j) = \begin{cases} a & \text{if } U_i^T V_j \leq 0 \\ a + U_i^T V_j & \text{if } 0 \leq U_i^T V_j \leq b - a \\ b & \text{if } U_i^T V_j > b \end{cases}$$

In our case, for classification problems we have the interval $[0-1]$ (we set a to a small value close to 0), and for regression problems $[0, 1]$ (also a to a very small value). To optimise the values of U and V we use gradient descent as follows: where μ is the learning rate that affects the learning time, a small value should be used such that the algorithm converges. The starting values of the algorithm are computed using the average value of all the scores,

$$U_{ij}, V_{ij} = \sqrt{\frac{\bar{R} - a}{f}} + n(r),$$

where f is the dimension of the SVD algorithm (the number of datasets and workflow

Algorithm 4 Complete incremental learning of SVD (adapted from [Ma, 2008])

Initialize matrices U and V
repeat
  for $\forall R_{ij}$ **do**
    Compute ∇_{U_i} and ∇_{V_j}
    $U_i \leftarrow U_i - \mu\nabla_{U_i}$, $V_j \leftarrow V_j - \mu\nabla_{V_j}$
  end for
until $RMSE$ starts to increase

features) and $n(r)$ is a random noise with uniform distribution in $[-r, r]$ to ensure that datasets and workflows have different features.

4.3.3.3. Other approaches

Naive Bayes The Naive Bayes classifier can be used to predict the ratings for new datasets based on the already executed workflows and the ratings provided by other datasets. During training the Naive Bayes requires two probabilities: the prior probability that the class label takes a certain value and the probability that a certain feature takes a value given that the class label is c. The probabilities are computed using the frequencies from the training matrix.

This method needs categorial values for the ratings. We have tested it only on discretized values of predictive accuracy and regression error. As the results are not very promising we do not present or discuss this method further.

Bayesian Networks In CF there are various methods that are based on Bayesian networks. The main idea is to identify a set of hidden features of users and items and use them to predict ratings for new users/items. This has been shown to provide good predictions. The approach requires to have categorial ratings. For our domain the ratings are real (predictive accuracy and regression error). The rating matrix needs to be discretized before applying this method. We tested one of the approaches (mixture of multinomials as described in [Marlin, 2004]) [Breese et al., 1998, Marlin, 2003] on a discretized matrix (accuracy is discretized according to its value in 10 intervals). However, the results were not better than the normal straightforward approaches from CF. For this reason we did not perform further investigations.

4.4. Content-Based Collaborative Filtering

The performance of classic CF is highly correlated to the sparsity of the rating matrix. To avoid this problem several approaches use additional information about the users and items. Some approaches employ this information to fill in the missing values. In addition to the rating matrix, for each dataset and workflow, we have a set of characteristics of both datasets and workflows. In this section we evaluate how this information can be used instead of classic CF. We devise some similarity measures based on the characteristics of the datasets and workflows. We start by introducing a list of features of datasets used in Meta-L and features of workflows that derive from the workflows' steps.

4.4.1. Features for Datasets

The work in Meta-L has explored various meta-features to improve the recommendation of the best algorithm for datasets. We base our feature selection on the approach of [Castiello et al., 2005] which summarises and describes features from previous approaches [Aha et al., 1992, Michie et al., 1994]. In these papers, three main categories of features are presented. We shortly introduce the most important features.

General features The following features describe general information about the dataset and are straightforward to compute (only by applying a function). These features are commonly used by many Meta-L approaches [Aha et al., 1992, Michie et al., 1994, Gama and Brazdil, 1995] and represent characteristics of the domain.

Number of examples is the total number of instances in the dataset (cardinality of the dataset, n_e).

Number of attributes represents the number of attributes or columns in the dataset (n_a).

Number of output values is the number of distinct values in the output column (the number of possible values for the target or label attribute).

Dataset dimensionality is the ratio between the number of attributes and the total number of examples in the dataset, i.e. $dim_{data} = \frac{n_a}{n_e}$.

Statistical meta-features Several statistics can be computed for numerical attributes of the dataset. Each statistic is first applied on every numeric attribute and then a mean is computed from the resulted values.

Standard deviation computes the dispersion of an attribute compared to its mean. If an attribute has n_e instances as $A = x_1, x_2, ..., x_{n_e}$ the standard deviation is computed as $std_A = \sqrt{\frac{1}{n_a}\sum_{i=1}^{n_a}(x_i - \overline{A})}$. Then the mean standard deviation over all attributes is computed as

$$\overline{std_A} = \frac{1}{n_s}\sum_{i=1}^{n_s}(std_{A_i})$$

Coefficient of variation represents the standard deviation normalized by the attribute's mean:$VarCoeff_A = \frac{std_A}{\overline{A}}$. For the entire dataset the average over all numeric attributes is computed:

$$\overline{VarCoeff_A} = \frac{1}{n_s}\sum_{i=1}^{n_s}(VarCoeff_{A_i})$$

Covariance expresses the linear relationship between two numerical attributes,$A_1 = x_1, x_2, ..., x_{n_e}, A_2 = y_1, y_2, ..., y_{n_e}$ and it is defined as:

$$Cov(A_1, A_2) = \sum_{i=1}^{n_e}\frac{(x_i - \overline{A_1})(y_i - \overline{A_2})}{n_e - 1}$$

, where $A_1 \neq A_2$. As a measure for a dataset one uses the mean of covariances for all different pairs of numeric attributes.

Linear correlation coefficient measures the linear association strength between two attributes. It is correlated to the covariance and computed as follows:

$$\rho_{A_1,A_2} = \frac{Cov(A_1, A_2)}{\sqrt{(std_{A_1} std_{A_2})}}$$

and its values are in the interval [-1,1]. Once again for the whole dataset the mean of all different pairs of attributes must be considered.

Skewness measures the asymmetry of an attribute. A positive skewness indicates that the tail on the right side is longer and a negative one the tail on the left side is longer. A zero value indicates that values are evenly distributed. Skewness represents the third moment of the distribution of an attribute A, divided by the third power of its standard deviation, $Skew = \frac{1}{std_A^3}\frac{\sum_{i=1}^{n_e}(x_i - \overline{A})^3}{n_e}$. The skewness of the dataset is the mean of the individual skewness of each numerical attribute.

Kurtosis describes the shape of the distribution of a given attribute in terms of peakedness. It is computed similarly to skewness, but instead of using the third moment one uses the fourth and then divides it with the fourth power of its standard deviation, $Kurtosis = \frac{1}{std_A^4} \frac{\sum_{i=1}^{n_e}(x_i - \overline{A})^4}{n_e}$

Information theoretic meta-features Information-theory provides measures that allow to find important attributes with respect to their class and can be used for both categorical or numerical attributes. Entropy is used to measure the randomness in attributes [Lindner and Studer, 1999b]. It applies mainly to categorical attributes but real-valued ones can be discretised and then their entropy can be computed as follows,

$$H(A) = -\sum_{i=1}^{n} q_i log_2(q_i)$$

, where $q_i = p(A = x_i)$ is the probability that the ith value of A is x_i and the sum is computed over all possible values of A from 1 to n. The values for entropy range in the interval $[0, log_2(n)]$, 0 meaning that there is a constant value and $log_2(n)$ meaning that values are uniformly distributed.

Normalized class entropy represents the information necessary to specify one class. The normalised class entropy is used in practice and is computed as

$$H(C)_{norm} = \frac{H(C)}{log_2(n)}.$$

Normalised attribute entropy computes the normalised entropy for each attribute and then the mean as follows

$$\overline{H(A)_{norm}} = \frac{1}{n_a} \sum_{i=1}^{n_a} (H(A_i)_{norm}).$$

Joint entropy of class and attribute computes the entropy of the combined pair (C, A), where C is the class variable and A one of the input attributes as $H(C, A) = -\sum p_{ij} log_2(p_{ij})$, where p_{ij} is the joint probability of observing the i-th value of attribute A and the j-th class value. The mean over all such pairs is considered as a measure for the overall dataset.

Mutual information of class and attribute measures the common information shared between the class attribute and normal input attributes. It can be computed as $MI(C, A) = H(C) + H(A) - H(C, A)$. The mean of all pairs is computed as an overall measure for the entire dataset .

Equivalent number of attributes indicates if the existing input attributes in a dataset are suitable to optimally solve the classification task. It can be evaluated by dividing the entropy of the class attribute by the mean of the mutual information between class and attributes:

$$EN_{attr} = \frac{H(C)}{\overline{MI(C,X)}}$$

Noise-signal ratio measures the amount of useless information contained in the dataset with respect to the relevant part. If we consider that the mean mutual information computes the amount of useful information about class then the irrelevant part is computed as $\overline{H(A)} - \overline{MI(C,A)}$. The ratio is computed as follows:

$$NS_{ratio} = \frac{\overline{H(A)} - \overline{MI(C,A)}}{\overline{MI(C,A)}}$$

To choose among all these features we have built the correlation table that finds high correlated attributes. We only keep attributes that have a correlation ratio less than a threshold (0.5 correspondingly -0.5, however the question remains which of the correlated features should be kept and which ones carry more information).

For regression problems we consider only the first two categories of features, general and statistical.

4.4.2. Features for Workflows

Each workflow consists of a set of steps that follow closely the CRISP-DM standard. We considered them as a potential discriminator for KDD workflows. This section assumes that the steps are specified in the HTN grammar and each of them represents a task as follows: *'CleanMissingValuesTask' , 'TypeConversionTask', 'NormalizeScalarTask', 'XValidationTask', 'AttributeSelectionTask', 'PredictiveModelingTask'*, and *'ModelEvaluationTask'*. The first step focuses on operators for cleaning the missing values from the data, next one converts different types of data such that they fit certain algorithms, the third one normalises the data. All the first three tasks are performing pre-processing and transformation of the data. The fourth step shows which cross-validation method is used for training the data (it could be skipped since it is the same for classification or for regression). The attribute or feature selection is done in the following step where attributes are selected based on their influence on the target attribute. The sixth step is the most important one since it selects the learning

algorithm to build a model. In the final step, different evaluation methods are used to generate and interpret the results.

Other features that could be used for further experiments are the average execution time of a workflow or the average CPU time.

4.4.3. Generating the rankings

The model generation was achieved using different similarity measures based on the features of datasets or workflows. First, we employ the reduced set of meta-features from the datasets for computing the similarity between datasets. For a new dataset, instead of executing workflows to find its cluster we extract its features and compute the cluster based on it. Given a dataset d_i, a set of features $f_{i1}, f_{i2}, ..., f_{in}$, $(n = 12)$, and a corresponding dataset d_j with features $f_{j1}, f_{j2}, ..., f_{jn}$ we compute the similarity of the datasets in two steps: First, we define first the similarity between features as follows [Kalousis and Hilario, 2003]:

$$sim(f_{i1}, f_{j1}) = 1 - \frac{|f_{i1} - f_{j1}|}{f_{1max} - f_{1min}}$$

Next, the similarity between two datasets is computed as the mean average of all features similarities:

$$sim(d_i, d_j) = \sum_{k=1}^{n} \frac{sim(f_{i,k}, f_{jk})}{n}$$

Then, to compute the rating/ranking of the dataset's workflows we employ the average rank of its neighborhood.

The second approach exploits the fact that the planner already informs us which workflows are applicable or not. Therefore, we can use this information to compute a similarity measure based on the applicability. To compute the distance between two datasets whose features are now binary data, where 0 means that the workflow is not applicable and 1 means it can be applied, we employ the Jaccard and Hamming distances. Both distances give the percentage of coordinates that differ, where Hamming distance is computed as

$$dist(d_i, d_j) = (\#(w_{ik} \neq w_{jk})/n)$$

and the Jaccard distance as

$$dist(d_i, d_j) = \frac{\#[(w_{ik} \neq w_{jk}) \cap ((w_{ik} \neq 0) \cup (w_{jk} \neq 0))]}{\#[(w_{ik} \neq 0) \cup (w_{jk} \neq 0)]}$$

The similarity is then computed as

$$sim(d_i, d_j) = 1 - dist(d_i, d_j)$$

Having the similarity between datasets, we can compute the ranking of a new dataset as the average of its most similar neighbors.

Another approach is to use the actual features of each workflow to compute the similarity between workflows. Instead of using the performance of each workflow one could use their structure to find clusters of workflows. Then, this information could be used to predict ratings for new workflows or for workflows which were not rated yet. For a new dataset with no executed workflows one could find an approximation of their performance by using the average rating from each cluster. This approach requires a large number of clusters with not so many workflows since otherwise we may end up with many workflows having the same rating.

We have extracted 5 relevant features of workflows consisting of the actual steps (without cross-validation operator and validation since they only differ by the type of problem they are solving). The features are relevant once we have a new workflow for which we have no ratings. The features allow computing the most similar items and using them as an initial prediction similar as for new datasets. But here the datasets are target attributes—a learner is built for each one of them. The same regression learners could be used as for new datasets. Workflow's features are also useful to find workflow clusters. An interesting point is to check if there is a connection between the clusters generated by the features and the clusters generated by the ratings in the matrix.

4.5. Hybrid Collaborative Filtering

We have described two approaches that are used in CF, pure CF that only uses the ratings and, content-based CF that only uses the content or information about datasets or workflows to compute similar datasets or workflows. The ratings in this case are used to approximate the ranks of new datasets on applicable workflows. However, one could combine both the information about datasets and the ratings to get an even more accurate similarity between datasets respectively workflows.

Approaches from CF research have proposed to use content-based CF to predict the missing ratings from the ratings matrix [Su and Khoshgoftaar, 2009]. A *pseudo-matrix* is built containing all the initial ratings plus the newly predicted ones via content-based CF. Then usual CF is applied on the new resulting matrix. In our

case we cannot fill the non-applicable workflows since they cannot be applied. But we can use this approach for the incremental learning part that drops a part of the ratings to test the influence of the sparsity on the CF methods (see Section 4.6.3). This solves the cold-start problem where for new datasets we can directly fill in some ratings without having to execute them. However, a mixed approached of execution and content is tested. First we use the same weights for both methods, and then once we have executed some workflows and we know the real ratings we can dynamically adapt the weights.

The question that arises is how to combine these to get a well-performing similarity measure such that the rankings of workflows are better estimated than using normal approaches.

Dataset similarity As described before we have two different sets of features that we want to combine: first the performance of workflows and then the meta-data. The main question that we are trying to answer is how to combine the two categories of features. The straightforward approach is to consider them as two different individual features and compute for each of them separately the similarity. Then a new similarity could be computed as the average of the two. Or consider them just as other features in the set so the total number of features would be the number of workflows plus the number of features and based on it compute the similarity. This would give the meta-data less weight. However, here the similarity between dataset needs to be computed separately for ratings and separately for the meta-data since there are missing ratings and the similarity measure is especially tailored for that. For that end we only need to focus on the right weights between these two similarities.

Workflow similarity The similarity between two workflows includes a new set of features that represent the operator steps. Similar to the previous approach one could test first the one where weights are 0.5 and then the one where the explicit features of workflows are considered just another feature.

4.6. Solving the cold-start problem

Having introduced our approach for applying CF to rank KDD workflows, we are left with finding appropriate strategies to choose suitable workflows for a newly arriving target dataset from the ones returned by the planner. To that end, we explored several strategies used in CF. In this sub-section we first present different statistics applied

on workflows and then consider information-theoretic approaches employed for cold-start problems in CF.

To improve the selection of workflows we consider the amount of precision improvement for a dataset if a certain workflow would be executed. As described in [Kohrs and Merialdo, 2001] this problem is reduced to identifying a set S of selected workflows from the set P of all applicable workflows for the dataset d_t that maximize the future prediction performance of the CF system for a target set T of workflows ($T \subseteq P$). This is further generalized as

$$\arg\max_{S \subset P}(perf_{\text{CF with } S}(T))$$

where $perf$ represents the performance of the CF system given the selected workflows S.

In the following, we succinctly describe some selection strategies. We differentiate between static smart strategies that rely on some property of the workflows and dynamic learning strategies that continuously evaluate which workflows to add to S. As a baseline we consider the random approach where the workflows for training are selected randomly.

4.6.1. Offline Strategies

The first approaches that tackle the cold-start problem in CF use statistic-based smart strategies [Kohrs and Merialdo, 2001, Rashid et al., 2002] to select initial items. These strategies explore the variance, entropy, popularity, and other variations of these measures shown in Table 4.2.

Workflows are selected either ascending by their *variance* across datasets (low variance signifies stability) or descending (high variance signifies more information about the dataset's peculiarities). Other approaches [Rashid et al., 2002] use *entropy* to differentiate between workflows. In [Kohrs and Merialdo, 2001] entropy is used to find which workflow better reveals the dataset's identity. The workflows are chosen in increasing order of entropy.

When several workflows with the same variance/entropy are selected they may be highly correlated/similar and therefore they do not offer new information. Consequently, the most correlated workflow to the selected set S can be removed and replaced with the least correlated workflow from the potential set P [Kohrs and Merialdo, 2001].

Furthermore, the entropy's definition excludes the missing ratings (or the non-applicable workflows) and also does not include the ratings' dispersion. To take this into account,

Measure	Description
Variance	$var(w_j) = \frac{\sum_{d_i \in D_{d_i,w_j}} (r_{d_i,w_j} - \overline{r_{w_j}})}{\lvert D_{w_j} \rvert}$ where D_{w_j} represents the set of datasets for which the workflow w_j was executed,and $\overline{r_{w_j}}$ represents the mean rating of all such datasets for the current workflow
Entropy	$H(w_j) = -\sum_{i=1}^{N} p_i log(p_i)$, where p_i represents the fraction of w_j's ratings which equal to i and N is the number of different ratings.
$Entropy_0$	$H_0(w_j) = -\frac{1}{\sum_{i=1}^{N} \alpha_i} \sum_{i=1}^{N} p_i \alpha_i \log(p_i)$ where α represents the weights for each rating category.
HELF	$HELF_{w_i} = \frac{2 * LF'_{w_i} * H'(w_i)}{LF'_{w_i} + H'(w_i)}$ where $LF'_{w_i} = \frac{log(\lvert w_i \rvert)}{log(\lvert D_{w_j} \rvert)}$, and $H'(w_i) = \frac{H(w_i)}{log(N)}$ where N are the number of ratings.

Table 4.2.: Measures used for selecting workflows in CF

Entropy$_0$ [Rashid et al., 2008] includes the non-applicable workflows with a smaller weight than for other rating categories (e.g., 0.5 vs. 1.0). Another variation of entropy is the *HELF* measure [Rashid et al., 2008] that combines the entropy and the logarithmic frequency of ratings. A different method worth mentioning is the popularity *Pop* that selects the most executed workflows and its combinations with the measures above such as *Pop*Entropy* or *Log(Pop*Entropy)*. Further simple approaches [Elahi et al., 2011] include the highest predicted (*HP*: workflows with the highest average ratings), lowest predicted (*LP*: workflows with the lowest average ratings), and a combination of both, where first the highest rated workflow is selected, followed by the lowest rated one, and so on.

4.6.2. Online Strategies

The methods presented so far select all training workflows S at once. Another category of methods first selects some workflows and only then learns which next workflows to pick. Information Gain through Clustered Neighbors (*IGCN*) [Rashid et al., 2008] uses the workflows' information gain [Mitchell, 1997] on the ratings of datasets that are most similar to the target dataset. It then clusters datasets based on their ratings to find the target dataset's profile. The approach is similar to the ID3 decision tree algorithm, where datasets' clusters are the classes and the goal is to build the

target dataset's profile by finding its right cluster. The information gain of a workflow can be computed similarly to the initial approach in ID3 [Harris Jr, 2002] as

$$IG(w_j) = H(C) - \sum_r \frac{|C^r_{w_t}|}{|C|} H(C^r_{w_j})$$

where, C is the distribution of datasets into clusters, $H(C)$ represents its entropy, and $C^r_{w_j}$ is the distribution of the datasets for which the workflow w_j was applicable and its rating value is r. The sum is the weighted average of the entropies of various partitions of the original class distribution (C) caused by the ratings for w_j for existing datasets. Since there are many non-applicable workflows for each dataset, this information helps figuring out the right cluster. Thus, information gain uses the weighted entropy :

$$IG(w_j; \alpha) = H(C) - \sum_r \frac{\alpha_r \frac{|C^r_{w_j}|}{|C|}}{E(C; \alpha)} H(C^r_{w_j}),$$

where

$$E(C; \alpha) = \sum_r \alpha_r \frac{|C^r_{w_j}|}{|C|}$$

and α are the weights.

The steps of IGCN are shown in Algorithm 5. In the initial selection step workflows are chosen based on their information gain (using all datasets). Then, the learning step considers only datasets from the target dataset's neighborhood to compute the information gain of the workflows. The algorithm should be applied only on categorical ratings (in our case we have predictive accuracy which can be from 0 to 1). For this reason the selection is done on a discretized matrix and not on the original one. The same is true for all entropy-based measures.

We modified the stopping condition for the second step such that the algorithm stops when it reaches a certain number of selected workflows.

[Harris Jr, 2002] observed that information gain favors attributes with many values. An alternative measure that overcomes this problem is the gain ratio [Mitchell, 1997]—now the default method for splitting attributes in ID3 [Harris Jr, 2002]—and employs the split information

$$SI(w_j; \alpha) = -\sum_r \frac{\alpha_r |C_{w_j}{}^r|}{E(C; \alpha)} \log(\frac{\alpha_r |C^r_{w_j}|}{E(C; \alpha)})$$

that is sensitive to how an attribute splits the data. We include the weighting of categories since we want to consider also the non-applicable workflows: $GainRatio(w_j; \alpha) =$

Algorithm 5 IGCN algorithm (adapted from [Rashid et al., 2008])

- Create c dataset clusters
- Compute information gain using all datasets

Selection step ▷ Select first ratings to build initial profile

- Select top n workflows with best IG score
- Add the selected workflows in the dataset's profile

end

Learning step ▷ Enrich the dataset's profile

repeat

- Find best l neighbors based on the dataset's profile so far
- Re-compute IG using only l datasets
- Select Top n workflows by their IG score (higher score first)
- Add selected workflows in the dataset's profile

until best l neighbors do not change

end

$\frac{IG(w_j;\alpha)}{SI(w_j;\alpha)}$. The split information discourages the selection of attributes with many uniformly distributed values. In our case, there are workflows with few rating values and, therefore, would be penalized by the IGCN method. We call the modified IGCN that replaces information gain with gain ratio *GRGN* (Gain Ratio Clustered Neighbors).

4.6.3. Incremental Learning

The presented approaches rely on a full matrix of ratings (actually in our case we treat the non-applicable workflows and the matrix is pretty sparse). However, in real life this is most often not feasible because it would require executing all the possible workflows for new datasets. This could be done offline if the resources were available but for many users this is not achievable. To circumvent this problem we test the impact of sparsity in the rating matrix for the best performing of the presented methods. For this reason we devise the following experiment: keep the training datasets (small ones) since executing them is not expensive, but for each testing dataset incrementally select only M workflows to be executed for training and then compute the prediction. The predictions for each dataset are based only on the previous dataset (a different approach than leave-one out), that is, an incremental approach. We are investigating how do the recommendations change with the sparsity. We test the hypothesis that the approach still works well even if some of the datasets from the matrix have ratings for a small number of workflows.

4.7. Discussion

We presented several CF methods in the context of ranking KDD workflows. Datasets are treated as users and workflows as items. For new datasets the ranking is computed by applying a CF method. As in other CF domains we are limited by the new item—workflow and new user—dataset problem. In this thesis we focus more on the new dataset problem and explore some strategies from CF research.

In this Section we introduced the foundation for solving RQ4. We showed how traditional CF approaches can be used to provide recommendations for KDD-processes. We discussed the memory-based, model-based and content based approaches. We also looked into solutions for the cold-start problem.

The next section presents an evaluation of the methods and hows how they perform in the context of datasets and workflows.

5 Evaluation

Several methods from CF have been introduced and adapted to recommend KDD workflows' ratings. This chapter describes the evaluation methods and presents the performance of the methods. First, we introduce the datasets used in the experiments for both classification and regression. Then, we continue with the experimental setup and present in detail the experiments and their setup. For the evaluation we identified a set of measures that are relevant for the experiments. Next, we report on the findings for each of the CF methods. We compare them and discuss their advantages and disadvantages. A set of limitations are discussed which identify some of the problems of the presented methods. Finally, we conclude with a short discussion about the overall evaluation.

5.1. Description of datasets

For testing the ranking of KDD workflows we selected a set of datasets from the UCI repository as well as from DVELVE, TunedIT, and KDD datasets online. The purpose was to build the case-base needed to employ CF and also to test the described approaches. We focus on two supervised learning problems: classification and regression. For each dataset we generated all the possible correct workflows with eProPlan and than executed them on a cluster.

5.1.1. Classification datasets

The classification datasets are solving prediction problems that either involve predicting a two-class target or a multiple-class target. We analyzed the rating distribution in

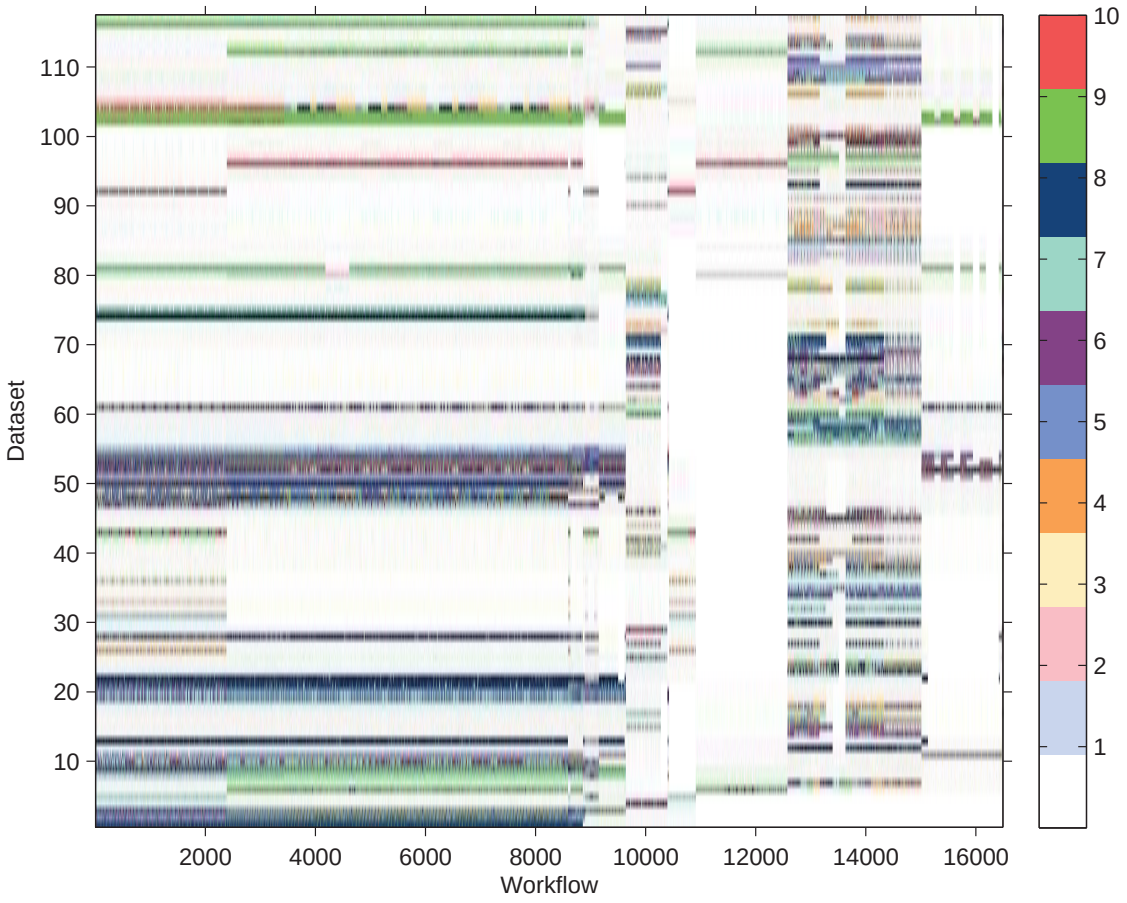


Figure 5.1.: Rankings of workflows for classification datasets (scale 1-10, where 0=not applicable, 1=worse, 10=best)

terms of predictive accuracy for each dataset and workflow to understand how workflows perform for different datasets. Fig. 5.1 illustrates how datasets rank certain workflows. To reduce the number of ratings/ranks and be able to visualize the entire matrix we discretize the ratings into 11 categories where 0=not applicable, 1=worst and 10=best. Looking at the color distribution we can observe that certain datasets have many workflows which are rated similar (quite homogeneous) especially workflows with ratings between 7 and 8. There are around 6 datasets with many good workflows (more than half) between 9 and 10 (the ones with green and red). There

are few datasets that have many bad workflows (between 1-4). This could be explained by the fact that trivial workflows were not generated (this information being modeled in the ontology). By only looking at the applicable and non-applicable workflows we can cluster the datasets into 4-5 categories. This only means that datasets in the same group have the same applicable or non-applicable workflows, but does not say anything about the performance of workflows. Another interesting statistic

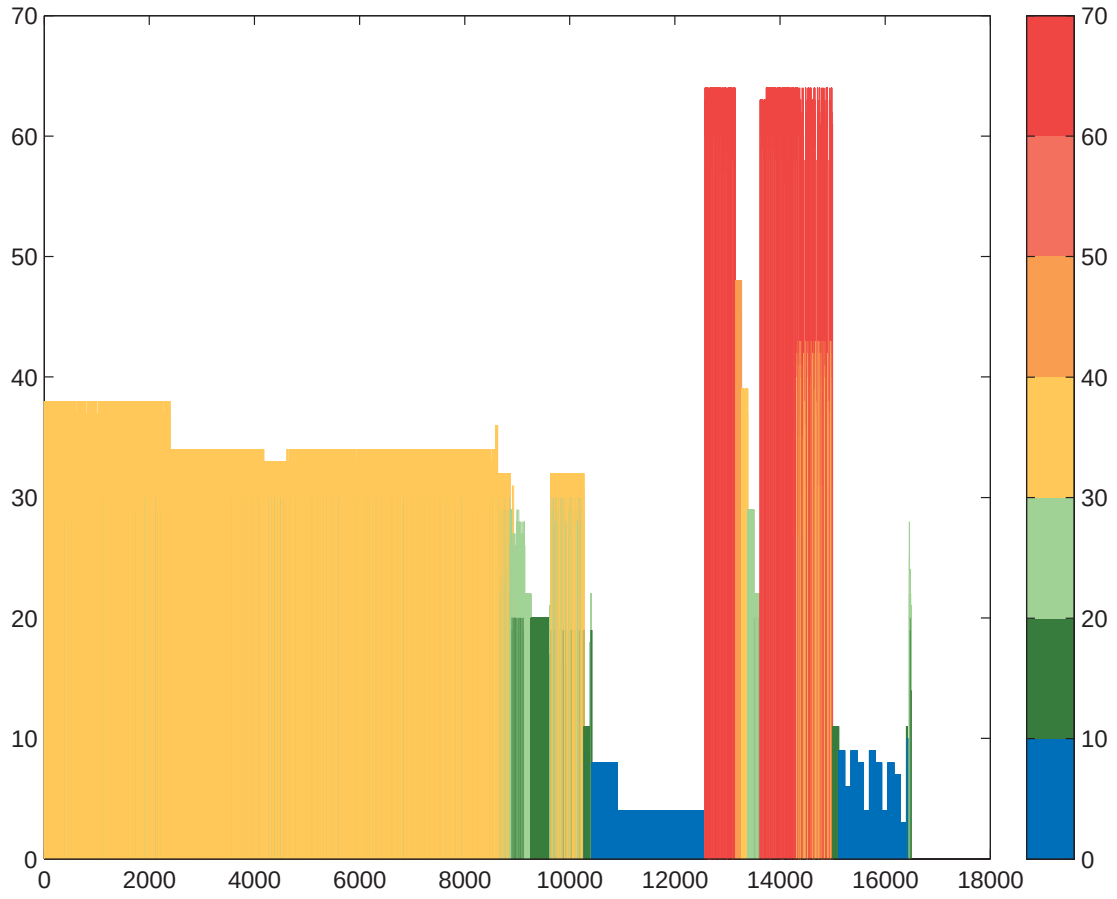


Figure 5.2.: How often are classification workflows applicable?

is the frequency of ranks for every workflow: 'How often is the workflow applicable?'. We present an overview of the frequency of usage in Fig. 5.2 There are around 4000 workflows which have been rated by few datasets (less than 10). However, a large number have been applicable for more than 30 datasets. Only around 100 workflows are rated by around 45 datasets. Looking at the rating frequency we can conclude that the frequency for more than average is acceptable. It would be interesting to see if there is a correlation between the rating frequency and the predicted ratings via CF.

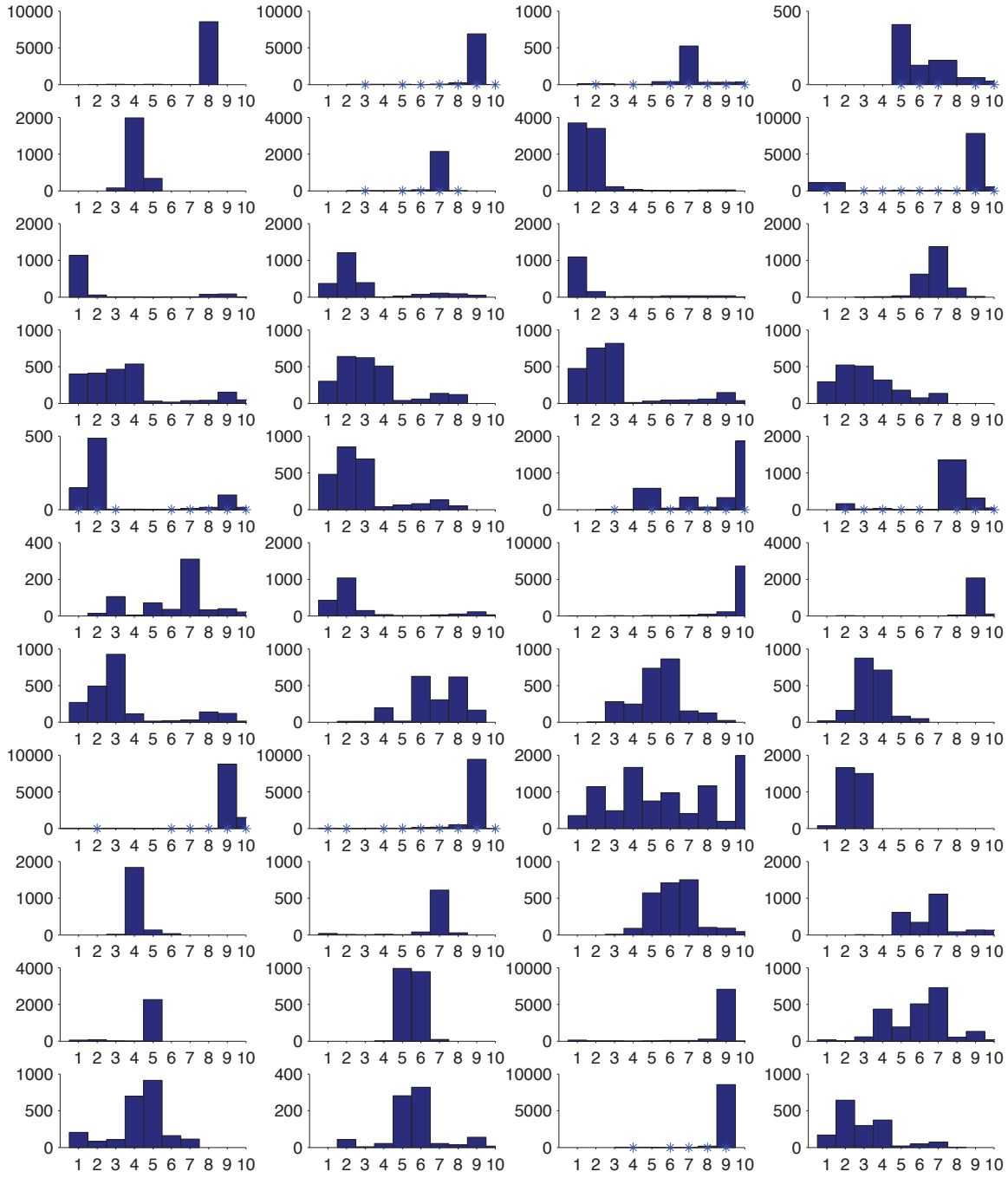


Figure 5.3.: Histogram of ratings for testing classification datasets (Y Axis - number of workflows, X Axis - Values from 1 to 10), datasets 74 to 117 from left to right

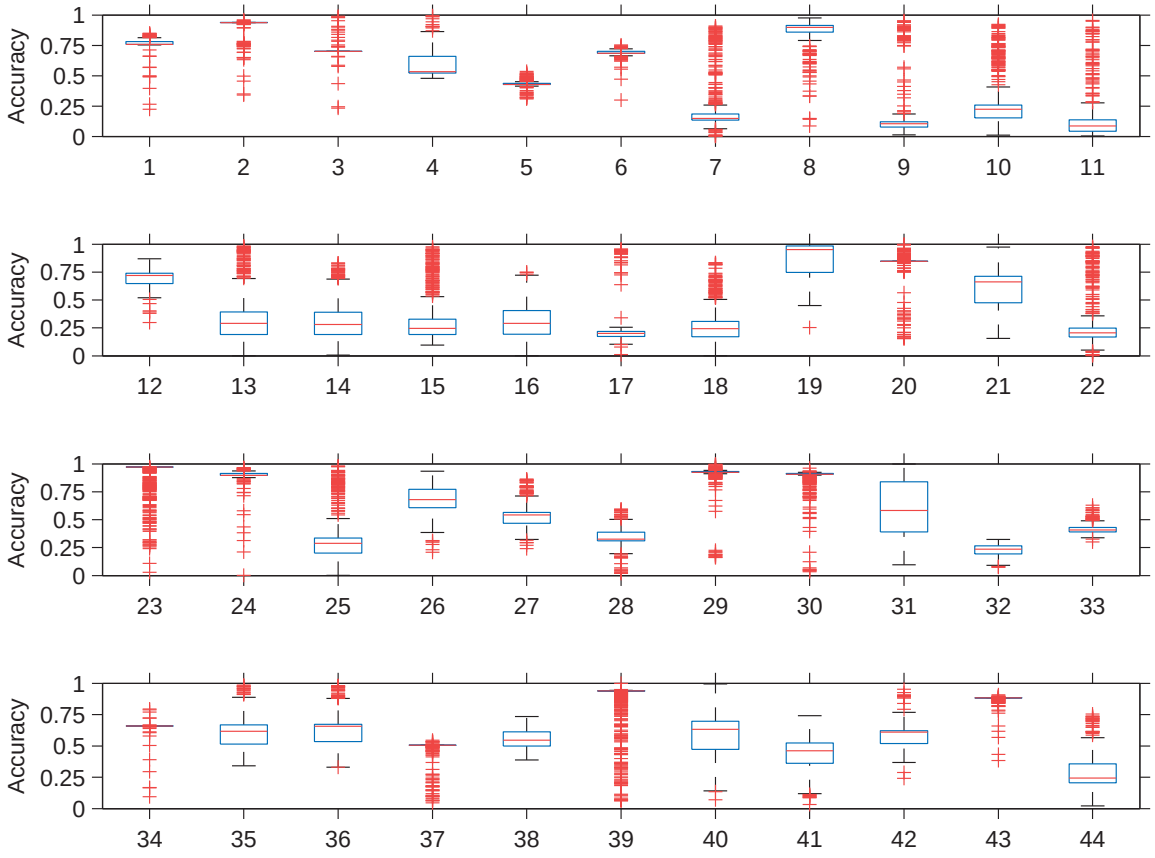


Figure 5.4.: Boxplot for testing classification datasets
(Y Axis - Accuracy, X Axis - Testing Dataset number)

For the test datasets (from 74 to 117) we have computed a histogram to understand how the ratings are distributed (from 1-10). As shown in Fig. 5.3 datasets have different minimum and maximum values. There are some datasets for which many of the best performing workflows are very close to the best overall workflow (e.g., 74, 75, 79, 81, 92, 96, 97, 102, 103, 104, 113, 116). The most interesting datasets are those that have either an uniform distribution or few good workflows. In our results' section we study closely the performance of CF methods on such datasets. One would expect that predicting the right ratings for such datasets is more difficult. To have a better impression about the ratings for each dataset we generated a boxplot for each dataset with the range, its mean rating and the standard deviation in terms of predictive accuracy in Figure 5.4. The distributions are quite diverse: a few dataset with lower mean ($mean < 0.5$), more with medium means ($mean > 0.5$, $mean < 0.75$), and some with high means ($mean > 0.75$).

5.1.2. Regression datasets

The regression datasets are handling numeric prediction problems where the target/label/class attribute is numeric. For our experiments we consider the NMAE (normalized mean absolute error) as a measure to compare workflows over different datasets. To have an idea of how workflows perform on these datasets we have discretized their performance values (the regression error can be from 0 to ∞) into two categories. The ratings are as for classification from 1-10 where 1 is the best and 10 the worse (better is when regression error is lower). As we have less regression learners in the ontology, the number of possible workflows for regression is much smaller than for classification (only 2000). Figure 5.5 shows the performance of workflows for all available datasets. In the case of regression most of the regression errors are between 0 and 10, even more precise between 0.9 and 1.5. But there are some cases that produce very large errors (larger than 1000). All those cases were converted to 1.0 which we consider to be the worst performance. That is why there are many workflows which perform relatively bad.

In Figure 5.6 one can see the number of applicable workflows per dataset. There are only a few workflows that are applicable for more than half of the datasets. Most of the workflows are applicable for only 15 datasets and only a few can be executed on less than 5 datasets. The question comes now how are the ratings distributed for each dataset. We are especially interested in the testing datasets. Figure 5.7 presents the number of ratings per category for each dataset. Contrary to the histograms for classification datasets, for most of the testing datasets we have a high

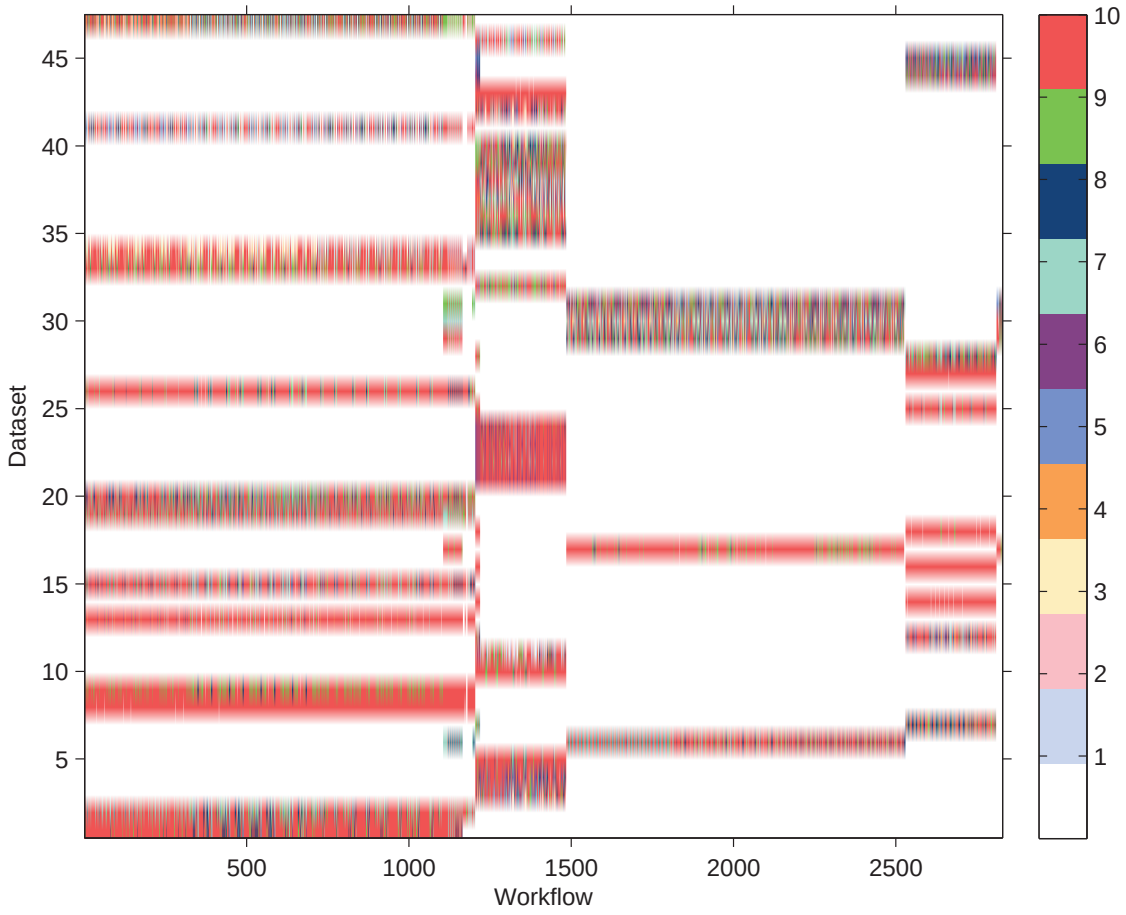


Figure 5.5.: Rankings of workflows for regression datasets (scale 1-10, where 0=not applicable, 1=best, 10=worst).

number of bad performing workflows (only datasets 36 and 37 have more good performing workflows). As such, in the evaluation section, we analyze closely those datasets for which this phenomenon is not present (datasets with uniform distribution of ratings, smaller number of very well-performing datasets, etc.). The boxplots for the test datasets are presented in Figure 5.8. This confirms the high number of bad performing workflows (0 - good, 1 - bad). The measure used is the relative absolute error where values above 1 have been replaced with 1 (workflows that are worse than the default model are not interesting).

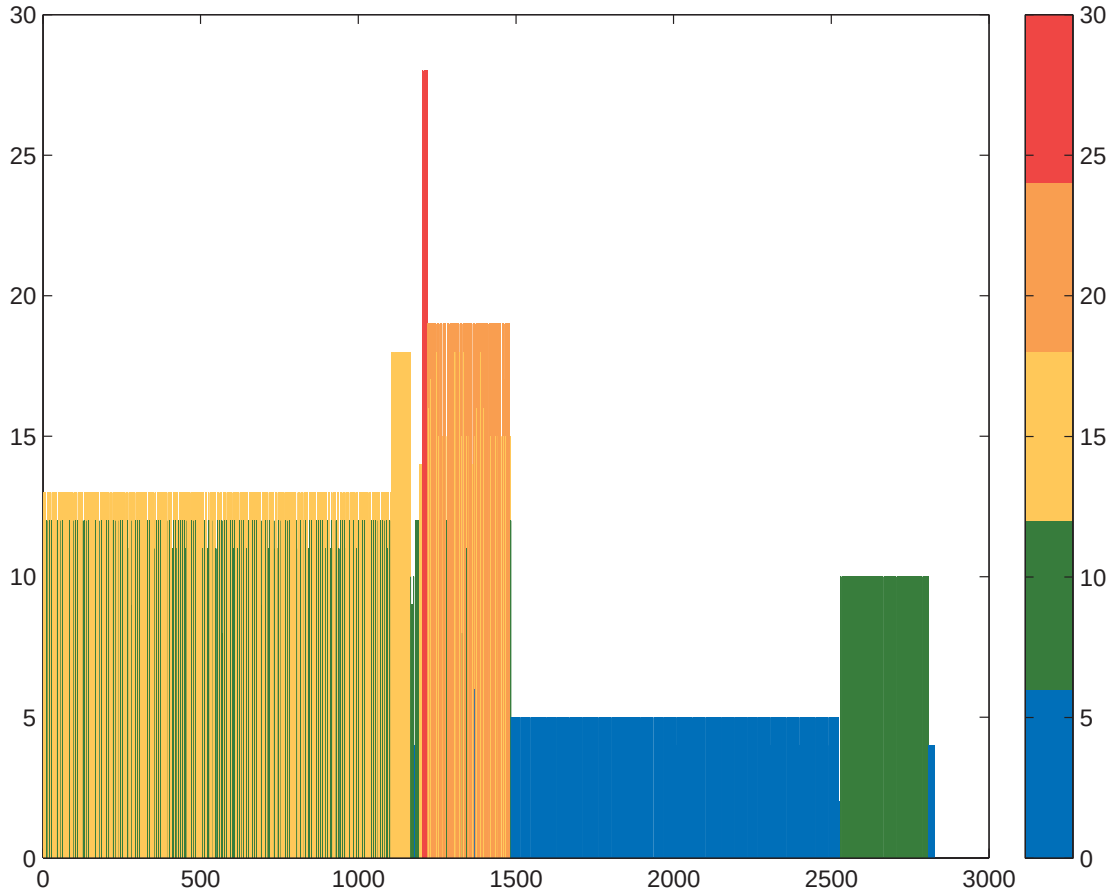


Figure 5.6.: How often are regression workflows applicable?

5.2. Experimental Setup

To build the case-base we selected the *classification datasets* from the UCI repository [Frank and Asuncion, 2010] and some from the TunedIT repository.[1] Using our eProPlan extension we generated all workflows for each dataset and executed them on a cluster. First, we filled the case-base in the form of the result-matrices $\mathcal{R}$ and I. To that end, we ran all workflows on all applicable datasets using 10-fold cross validation and then uniformly discretized the resulting accuracy into 10 different categories (1-10) to ensure comparability between datasets. Second, we ran all tests using only the *target datasets* with more than 1000 instances (datasets with < 1000 examples we call *background datasets*). To avoid overfitting those datasets we split them into training and testing (70%-30% or their initial distribution from UCI if any).

[1] http://tunedit.org/

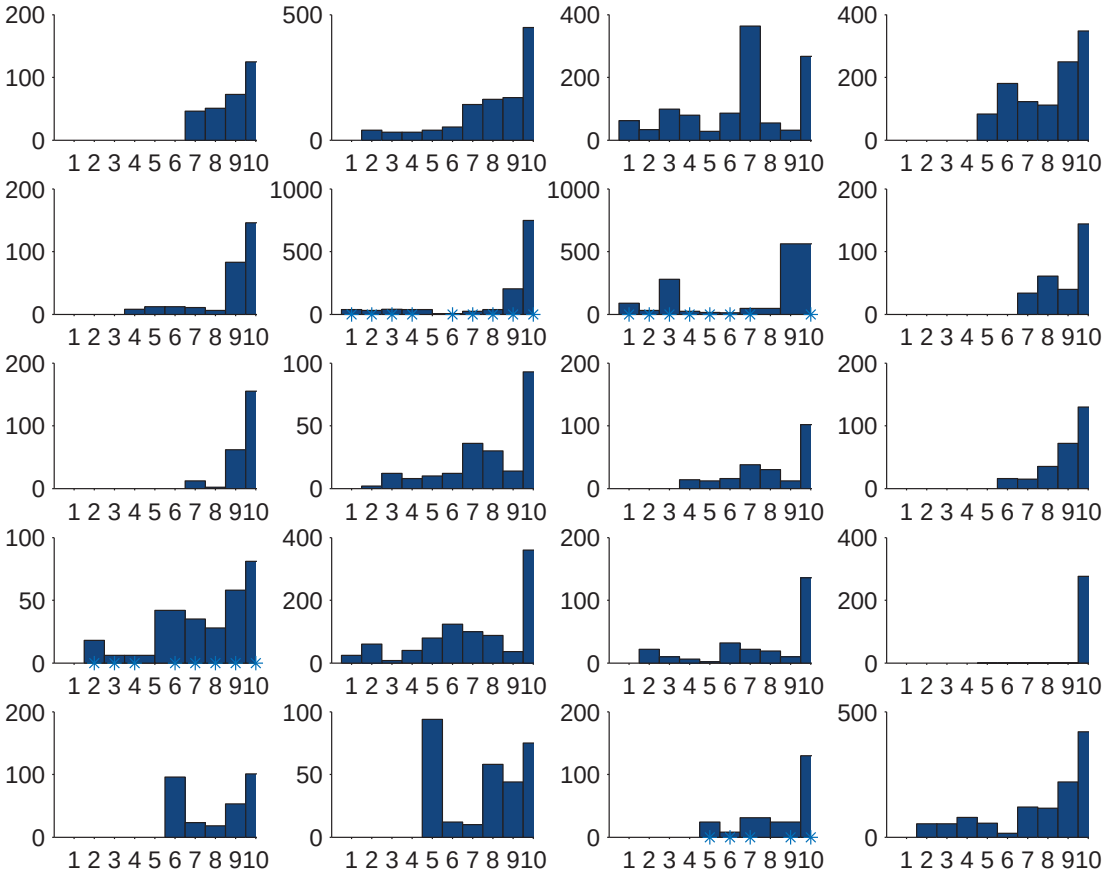


Figure 5.7.: Histogram of ratings for testing regression datasets (datasets 27 to 47 from left to right)

The training part was used to generate the rankings for the workflows; the testing part only to validate the predictions.

To evaluate our rankings we used the steps shown in Fig. 5.9. In all our experiments we employed the *leave-one-out* method to ensure a strict division between predictions and background information. Predictions are made by first selecting the training ratings for the target dataset using one of the strategies described in Section 4.6; then kNN is used to compute the missing ratings based on similar neighbors. As discussed, our choice mechanism has several dimensions (see Table 5.1). For the dataset-based approached all the neighbors are considered. For the workflow-based approaches we chose k=10 (if possible) as they were more expensive to compute (we have more workflows than datasets). We tested several variations of the presented methods as follows: variance ascending or descending (*VarAsc*, *VarDesc*) with or without optimization ($VarAscOpt_{10}$ uses 10 steps for the optimization), entropy

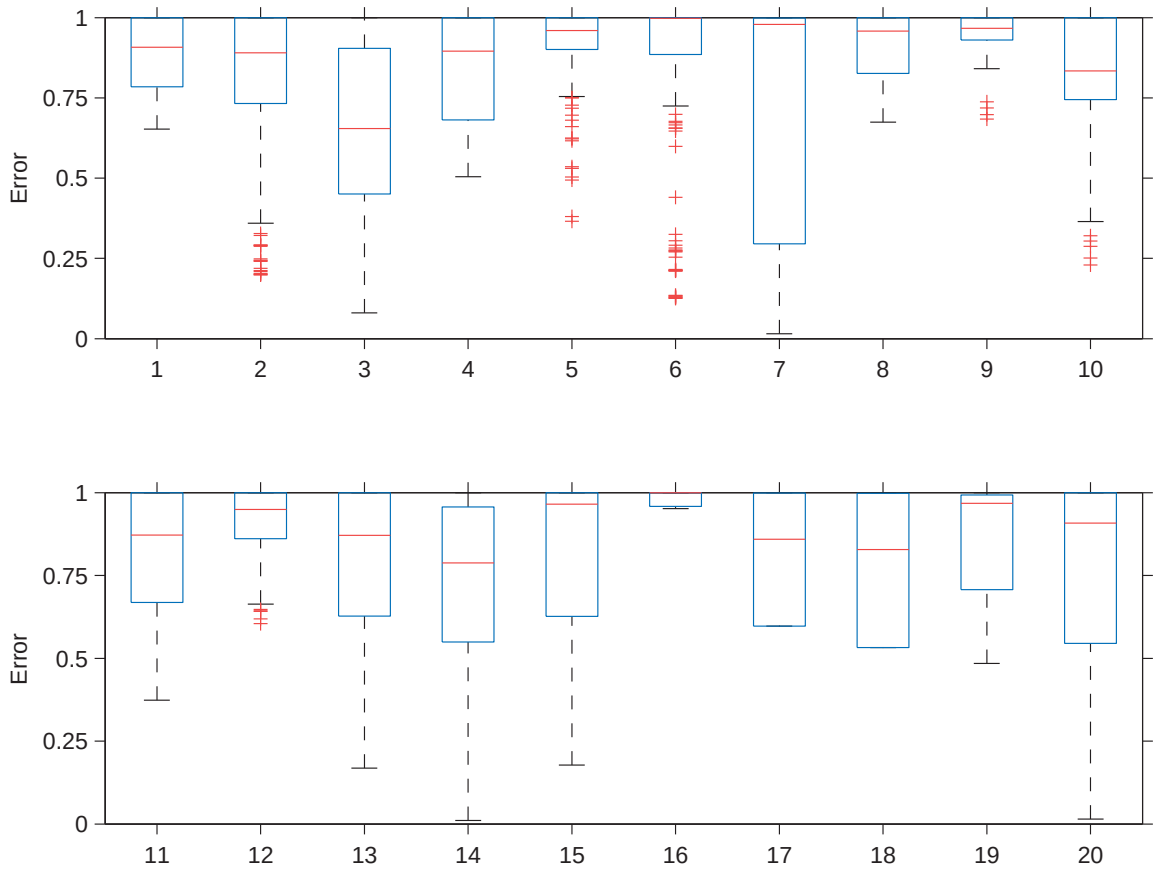


Figure 5.8.: Boxplots for testing regression datasets
(Y Axis - Regression error, X Axis - Dataset)

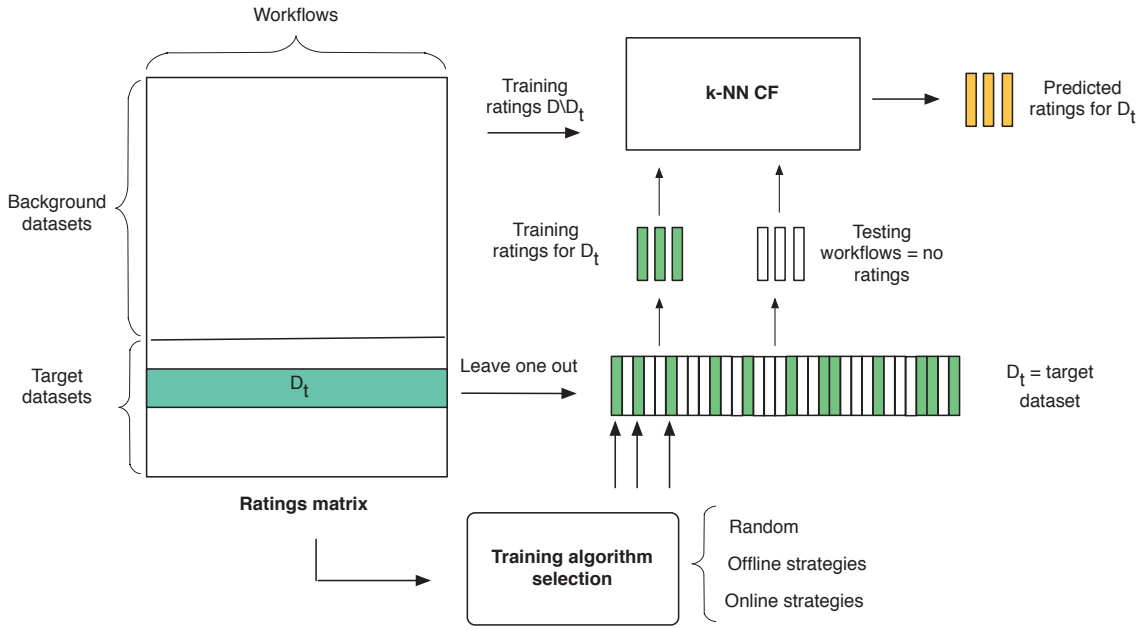


Figure 5.9.: Experimental setup

as defined by Kohrs (*EntKAsc* or *Desc*) or classic entropy (*EntAsc* or *Desc*), and Ent_0 that considers non-applicable workflows as category 0. We also tried statistics like popularity-based *Pop*, most/least used (*HP*, *LP* and *HPLP*), *PopEnt*, *logPopEnt*, and *HELP*. As learning strategies we tested *ICGN* and *GRCN* with their variations (with entropy, Gini Index and classification error).

5.3. Evaluation Metrics

In this section we discuss the evaluation metrics used for assessing the quality of workflows as well as the quality of CF recommendations. First, we take a look at the classic measures for ML algorithms in the field. As measurement for the quality of the predictions we adopt metrics from CF. In addition, we use a best in TOP-K workflows approach to measure how good the predicted workflows are in comparison to the best real workflow.

5.3.1. Metrics for KDD workflow evaluation

Our approach considers only supervised learning—classification and regression tasks.

Classification tasks The goal of classification problems is to predict a non-numeric attribute (nominal or ordinal) based on the other attributes. The learning uses one part

CF-Method	Similarity-Method	SelTrain Method	SelTrain Algorithm	# training workflows
Dataset-based	PC	Baseline	Random	5
	Vector	Statistics-based	VarDesc	10
	PIP		EntDesc	
			Pop	15
			HP	
			LP	25
			HPLP	
Workflow-based	PC		PopEnt	50
	Cosine		LogPopEnt	
			HELF	75
	ACosine		Ent_0	
		Learning-based	IGCN	100
			GRCN	

Table 5.1.: Explored methods for selecting workflows

of the data and builds a model that is then tested on the remaining part. Common metrics for classification tasks are: predictive accuracy, precision and recall, f-measure, Area Under Curve (AUC). The measure used in this thesis is predictive accuracy. For future work we may consider using also other metrics.

Regression tasks Here the predicted class is numeric and the idea is to measure the error between the real class value and the prediction. The most common measures are mean-squared error, mean absolute error, relative squared error, relative absolute error and correlation coefficient. In our experiments we have considered the Root Mean Squared Error (RMSE), Absolute Error (AE), RelativeError (RE), and Normalized Absolute Error (NAE). These measures where computed using the RapidMiner Performance operator for regression. To compare the performance of workflows across datasets we have used the relative or normalized absolute error. This represents the absolute error divided by the error of the simple predictor that predicts average values also called relative absolute error [Witten and Frank, 2005]. A similar measure, normalized mean squared error is used to compare regression learners across several datasets [Köpf et al., 2000] [2]. In our results we are going to refer to it as NAE or simply regression error.

[2]The RM operator that measures the performance of regression algorithms only offers the NAE measure.

5.3.2. Metrics for CF prediction evaluation

As a first measure we employ *Mean Absolute Error* (MAE), defined as the average between the actual ratings and the predicted ones.

$$MAE = \frac{\sum_{t,i} |r_{t,i} - p_{t,i}|}{N}$$

Here $r_{a,i}$ is the actual rating of the target dataset t and workflow i, $p_{t,i}$ is the predicted value for it and N is the total number of ratings in the test set. The lower the MAE is the better the CF-method is.

5.3.3. Metrics for Workflow evaluation

Similar to the recommendations from search engines, we provide suggestions on which workflow performs best on a given dataset. When using a search engine a search is successful, if a good match is visible on the first page. Analogously, we employ a *best in top K predicted* metric. This metric selects the top K recommended (or predicted) workflows (in terms of their ranks), checks their actual accuracy, and compares the best of the actual accuracies to the best real workflow's accuracy. In essence, we measure how well compared to the best possible workflow a user would fare, who would choose the best performing workflow (based on some test dataset) from those K top predicted ones. We denote *accuracy loss* as being the mean of the differences of accuracy to the top best workflow.

Another metric is the coverage (precision, recall): how many of the real TOP N are in fact in the real TOP N? We want to find out how many of the top workflows we are able to predict correctly.

5.4. Selected methods

For comparing the quality of KDD workflow ratings in terms of CF we have explored four different categories of CF methods: classic CF, model-based CF, content-based CF, and hybrid CF.

5.4.1. Classic CF methods

Several methods from CF are combined with different selection strategies for the training workflows. Two categories of methods are used depending on how the similarity

between the target dataset and the rest is computed (dataset-based vs. workflow-based). For all methods we use k-NN ($k = 10$) with different similarity measures and selection strategies.

Dataset-based methods are using the similarity between the target dataset and the K most similar datasets to compute the neighborhood or cluster of the current dataset as explained in Section 4.3.2. We are comparing several similarity measures (Cosine, PC, and PIP) and 3 selection strategies (Random, Pop, HELF, and IGCN) (see Section 4.6). This gives us the following methods: *DS CosineRandom*, *DS CosinePop*, *DS CosineHELF*, *DS CosineIGCN*, etc. For the online strategies we only keep the entropy-based one (IGCN), since the other variations performed worse.

Workflow-based methods use the similarity between workflows to predict the ratings for the target dataset. As for dataset methods we have as similarity measures (Cosine, PC, AdjustedCosine) and a separate method SlopeOne, combined with the 3 selection strategies.

5.4.2. Model-based CF methods

For model-based methods we compare the kMediansClustering and the SVDClustering both introduced in Section 4.3.3. For kMedianClustering we usee 3 selection strategies (Pop, HELF and IGCN). We also test the incremental learning SVD (SVDInc). The parameters used in the experiments for classification datasets are: $k_d = 0.025$,$k_w = 0.025$, $\mu = 0.005$, the a value added at each step is 0.1 (a smaller values leads to more steps for the gradient descent algorithm) and we limited the number of steps to 100. For regression we used the following parameters: $k_d = 0.05$,$k_w = 0.05$, $\mu = 0.01$, the a value added at each step is 0.015 (a smaller values leads to more steps for the gradient descent algorithm) and we limited the number of steps to 250.

5.4.3. Content-based CF methods

For datasets we use the MD method based on a selection of features and for workflows the Bin method based on the presence of applicable workflows for each dataset. These two methods are using features of datasets and workflows to compute the similarity between the target dataset and the case-base. The ranking is then computed similar as for kNN methods. Additionally, we test several regression algorithms from Matlab where the datasets features are the predictive attributes and each workflow

is the target attribute (class). The algorithms that we have considered are k-NN (k = 10), Linear Regression and RegressionTree.

5.4.4. Hybrid-based CF methods

We combine some of the CF with the content-based ones. We make use of the content-based methods to fill in missing values. This is only a beta-approach and remains to be improved. For a target dataset with no information other than its meta-data we predict the ranks of its applicable workflows using a good performing content-based CF. This allows us to find k most similar datasets and use a CF method to predict the rankings. A second approach uses both CF and content-based one to predict the ratings and then computes a final rating (equal weight 0.5).

5.5. Experimental Results

In the following experiments we are exploring: (a) first how to recommend a workflow ranking for new datasets, (b) second how to recommend the best performing workflows, and (c) how does sparsity influence the recommendations. For this purpose we show the performance of the previous described methods.

As selection methods for the training workflows we only present the most successful methods, Pop, HELF, IGCN and Random. In the following we present the performance of the above methods in terms of Loss of MAE and difference to the best real workflow.

Classification Problems

The experiments employ 117 classification datasets, where 73 are only used for training and have a low number of examples (instances). The rest 44 datasets represent the testing datasets. We use the leave-one-out approach to test the performance of the CF methods. For each testing dataset we use the performance on the training part to select the training workflows for CF and then the test part to asses its performance. This ensures that we do not overfit the data (not using the same data for building the model and then testing it).

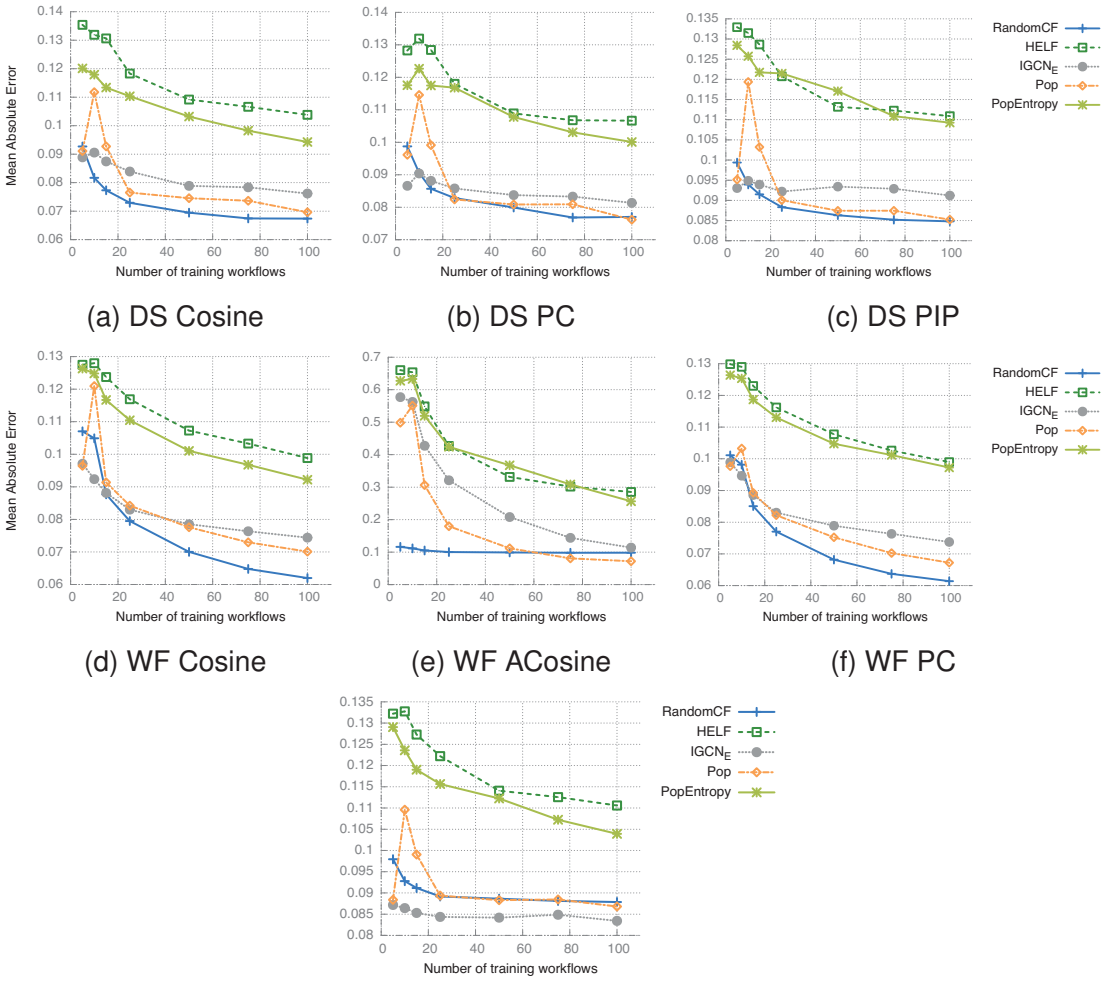
Regression Problems

For regression we have used 47 datasets: 27 only for training in the case-base and 20 datasets for testing. We used the same leave-one out approach as for classification.

5.5.1. Memory-based CF

We present the performance of the previous methods and algorithms in terms of their CF prediction as overall MAE but also per dataset. The MAE is computed for predictive accuracy respectively NAR.

For the second experiment we used predictive accuracy as a rating. For IGCN we employ squared euclidean distance for k-means and for computing the neighborhood (with 10 replicates).



(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC

(g) WF SlopeOne

Figure 5.10.: MAE for Dataset and Workflow-based CF (classification task)

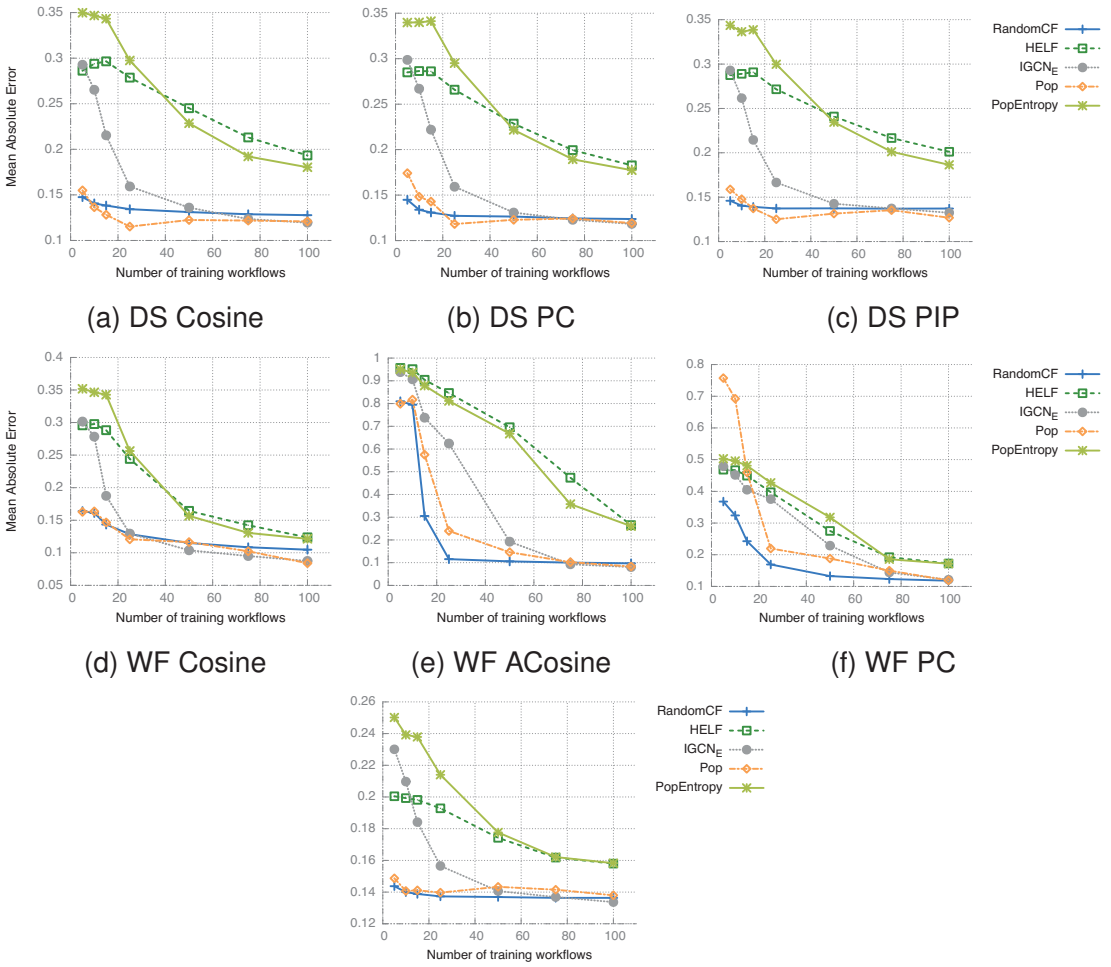
For recommending a good workflow we employ two measures: first, best of TOP N and second, coverage – how many from the predicted TOP N are in the actual TOP N.

Figure 5.10 shows the improvement of MAE with the number of selected workflows for training. The best performance is achieved by IGCN followed closely by random selection with CF and Pop. In some cases (DS PC, WF Cosine, WF PC) IGCN is better when the number of workflows is smaller (5, 10). For SlopeOne the online approach outperforms the others.

For regression problems, when using NAR as a rank, we have the situation when the error is for some cases very high. This shifts the overall MAE to a higher scale. For this reason we present the results using a log scale. As shown in Figure 5.11 the total MAE is very high (due to the few bad performing workflows with high regression errors). We can see that in most of the cases the intelligent strategies for choosing the training workflows perform much better that the random approach. Also the methods improve with the number of workflows selected for training except in some cases (PIP and SlopeOne have interesting variations). These may be due to the size of the case-base and the number of datasets used for testing that is less than half of the one for classification problems. Another problem that we have encountered is in the case of workflow-based CF. Here, we use regression to predict the ratings of the neighbors. In the case of regression the number of common rated items is relatively small and the regression does not work well. We replaced it with the normal prediction that only considers the neighbors, it improves but it is not as good as the dataset-based methods.

To determine the quality of the predictions compared with the best performing workflow, we employ a measure called the distance to the best real workflow. We take Top1, Top 3, Top 5 and Top 10 predicted workflows and compare it to the best real ones.

For classification, the number of applicable workflows goes from 819 to 11230. Recommending the user Top 10, executing them and getting at least one very good workflow it is already an improvement. We consider the recommendations starting from Top 1 until Top 10. Executing more than 10 workflows, in our opinion, will be in some cases too costly (for large datasets). For regression, the statistics are different: datasets have from 200 to 1200 applicable workflows. Executing 10 workflows would make it easier to find a better workflow. That is why for regression problems we only consider Top 1, Top 3 and Top 5. We also compare with a simple random method that select N workflows at a time, executes them and selects the best in terms of predictive accuracy or regression error. This shows how better the CF ranking methods are compared to the trivial choice method. The random approach was computed for both tasks as a mean of 100 runs. This is only an approximation of a real random since



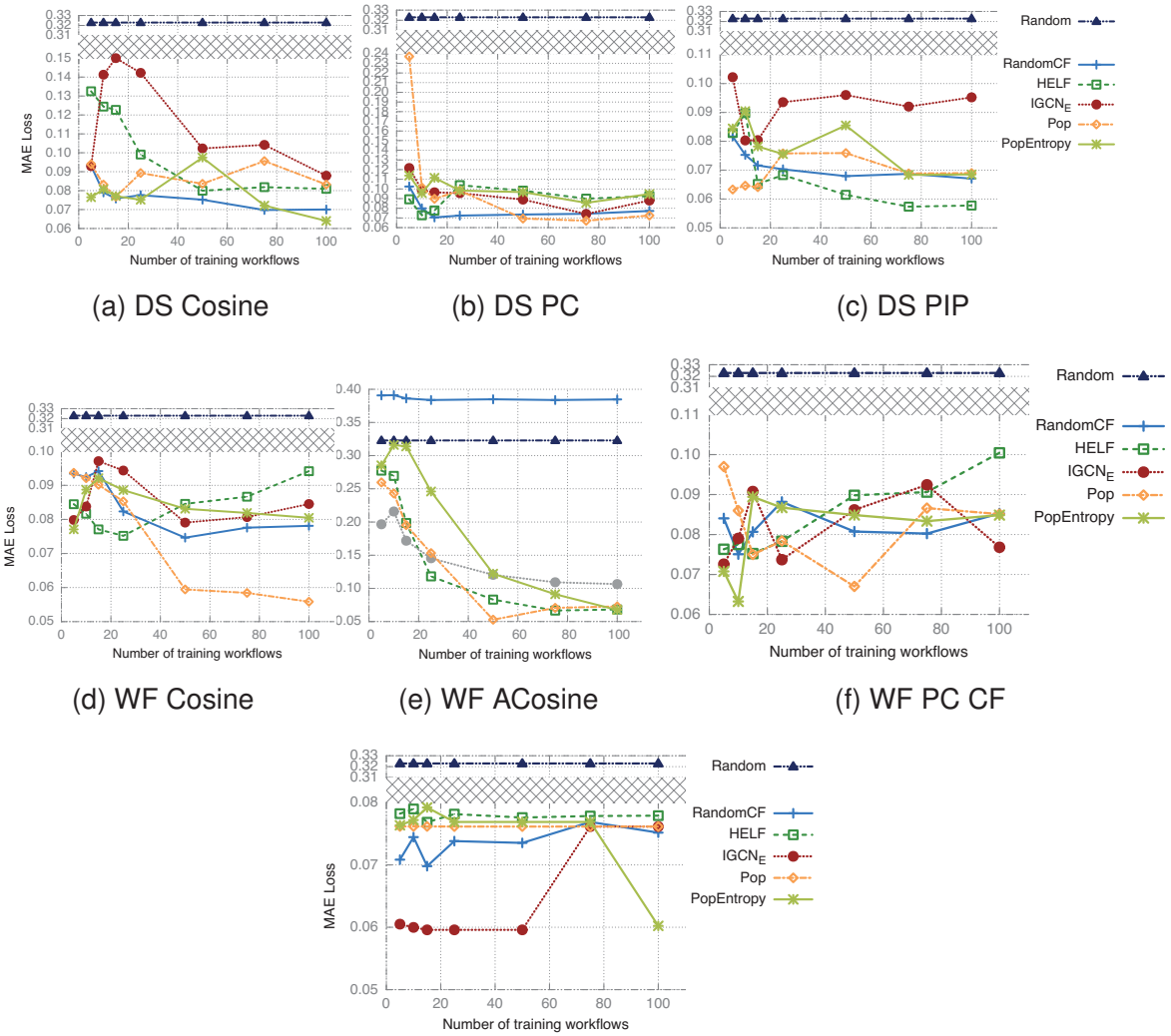
(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC

(g) WF SlopeOne

Figure 5.11.: MAE for Dataset and Workflow-based CF (regression task)

not all the possible values are covered (e.g., in the case of regression one may get very bad regression values that influence the overall mean).



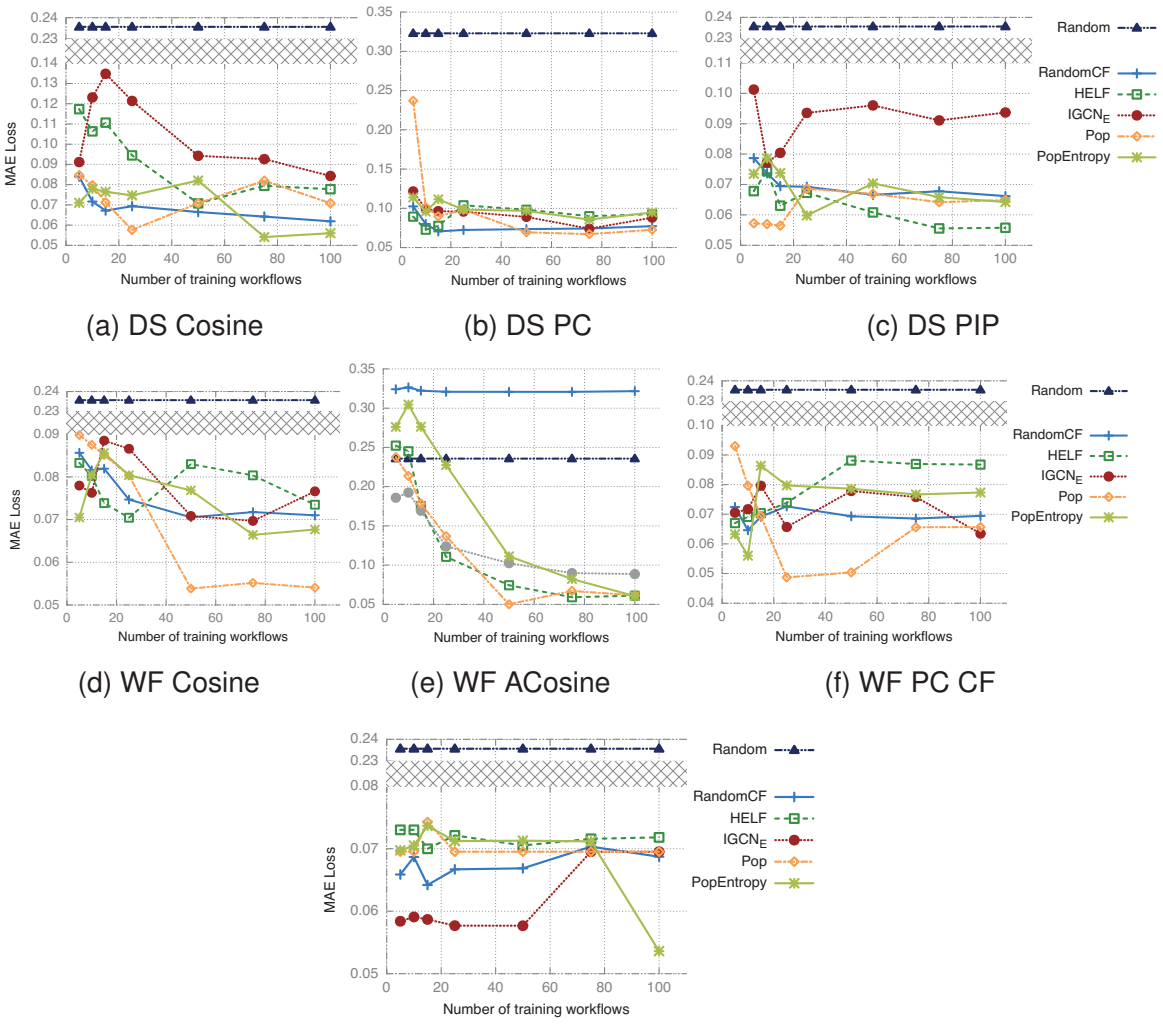
(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC CF

(g) WF SlopeOne

Figure 5.12.: MAE Loss between Top1 predicted and best real workflow for Dataset and Workflow-based CF

The recommendation for Top 1 for classification tasks (see Figure 5.12) improves with the number of workflows and is significantly better that the trivial random strategy. HELF with PIP selection and IGCN with SlopeOne show better performance than the other approaches. A slighter advantage is also shown by the Pop with PC. The Random approach performs constantly well with all the similarity measures. But we should keep in mind that it contains the mean of 100 runs. All the other selection

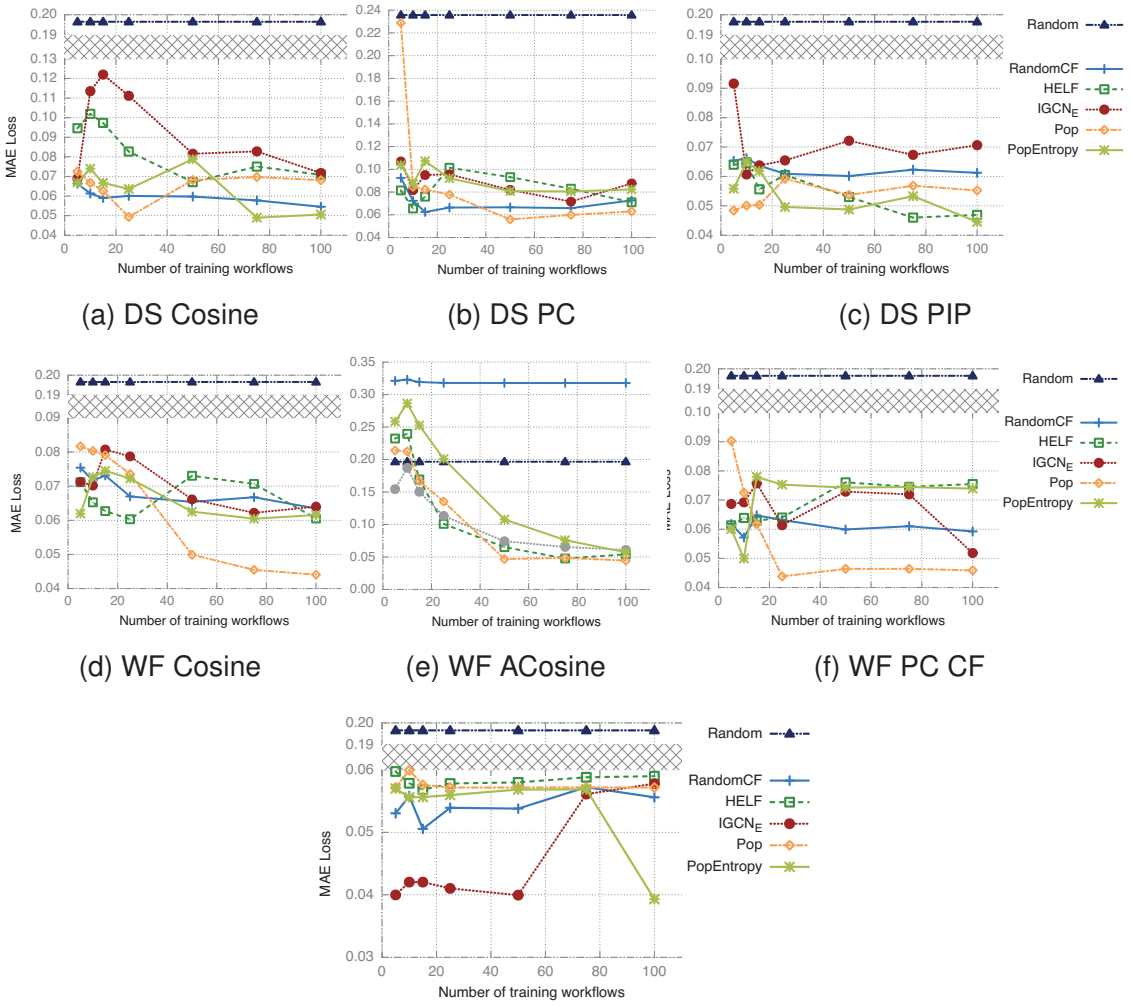


(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC CF

(g) WF SlopeOne

Figure 5.13.: MAE Loss between Top3 predicted and best real workflow for Dataset and Workflow-based CF

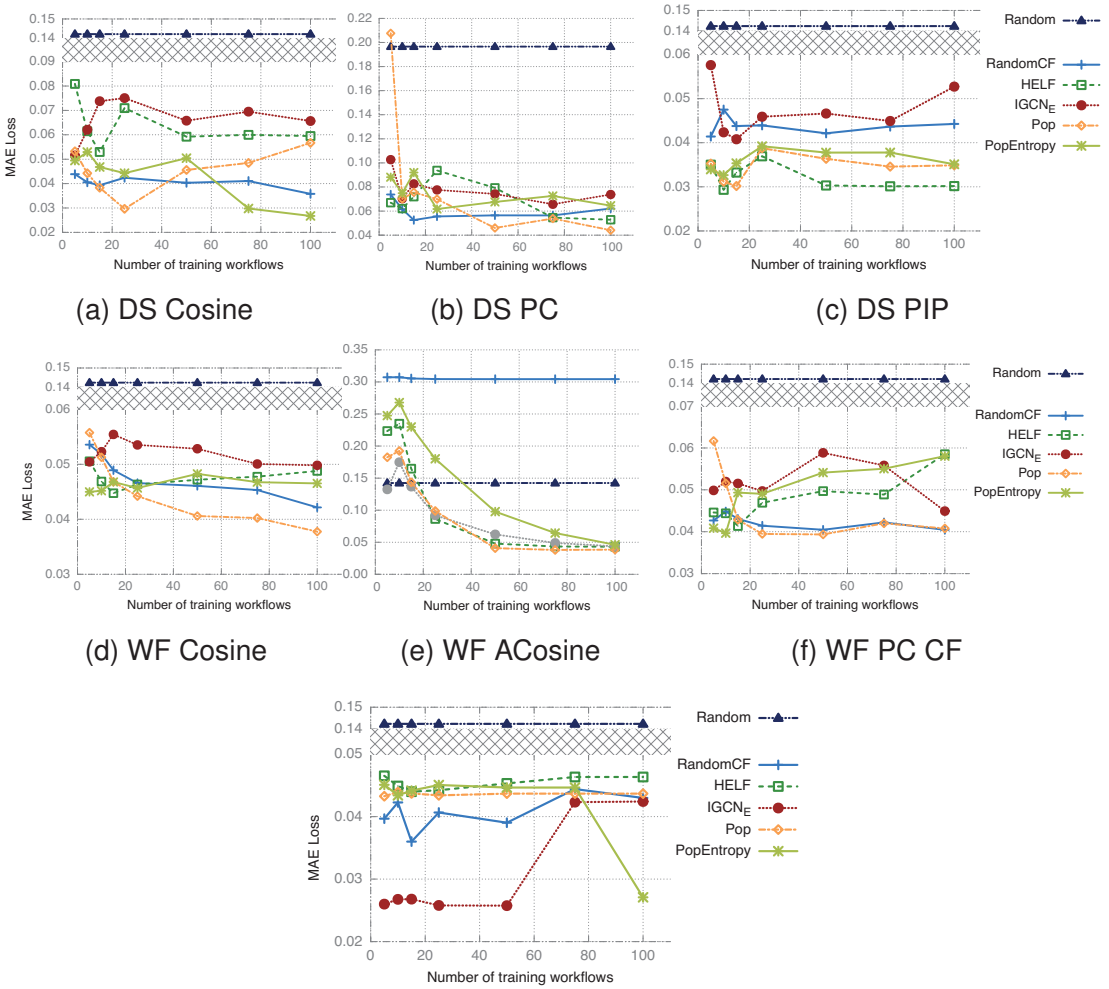


(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC CF

(g) WF SlopeOne

Figure 5.14.: MAE Loss between Top5 predicted and best real workflow for Dataset and Workflow-based CF



(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC CF

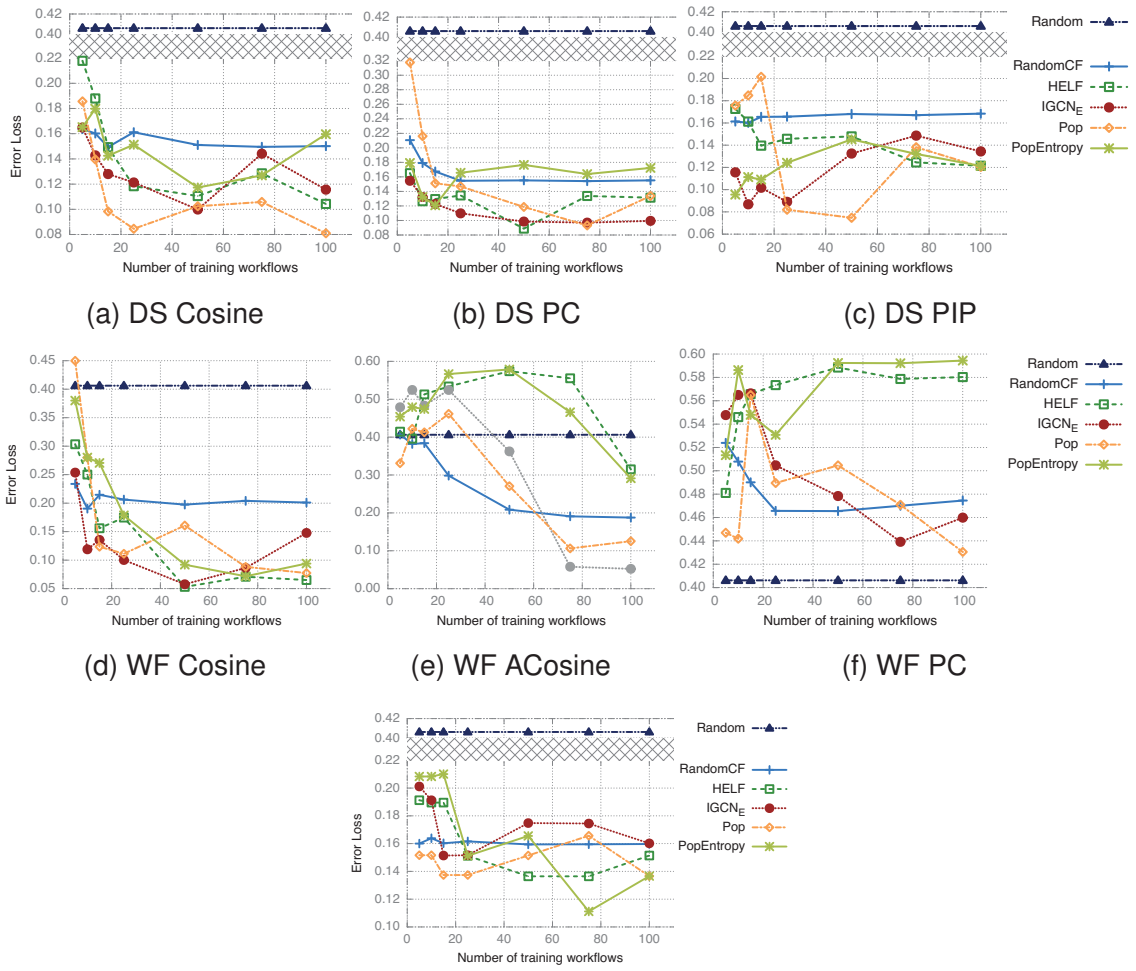
(g) WF SlopeOne

Figure 5.15.: MAE Loss between Top10 predicted and best real workflow for Dataset and Workflow-based CF

strategies are deterministic and provide a comparable and in some cases even better performance.

The loss in error decreases from 0.1 for Top 1 to 0.05 and even 0.03 for IGCN with SlopeOne for Top 10 (Figure C.2). For a relatively good prediction one could use one of the strategies for selection with 10-15 training workflows.

For regression tasks the performance of CF methods is slightly worse. This is due to the fact that some workflows have relatively few ratings. In terms of error loss to the best real workflow DS PIP, WF SlopeOne and DS Cosine they perform better. The workflow-based approaches (with Cosine and PC) perform quite bad.

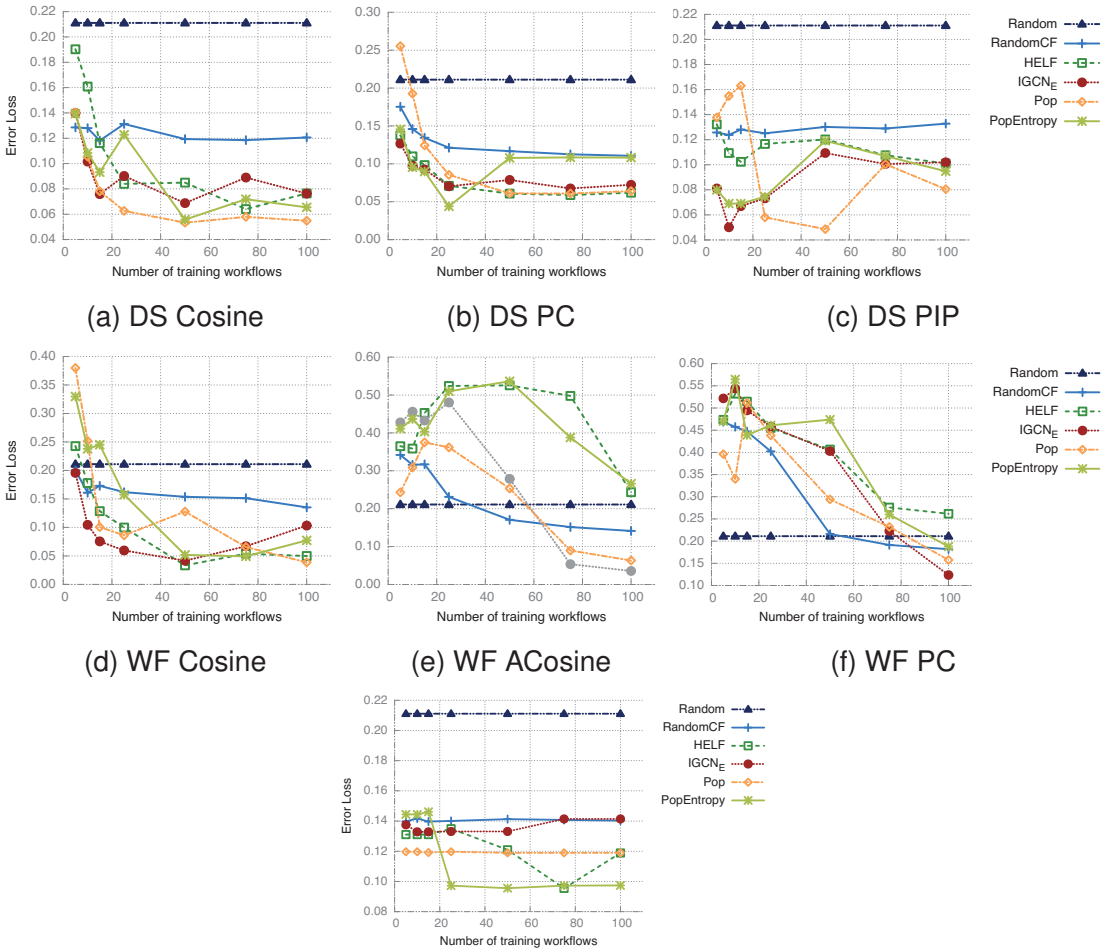


(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC

(g) WF SlopeOne

Figure 5.16.: NAR Loss between Top1 predicted and best real workflow for Dataset and Workflow-based CF

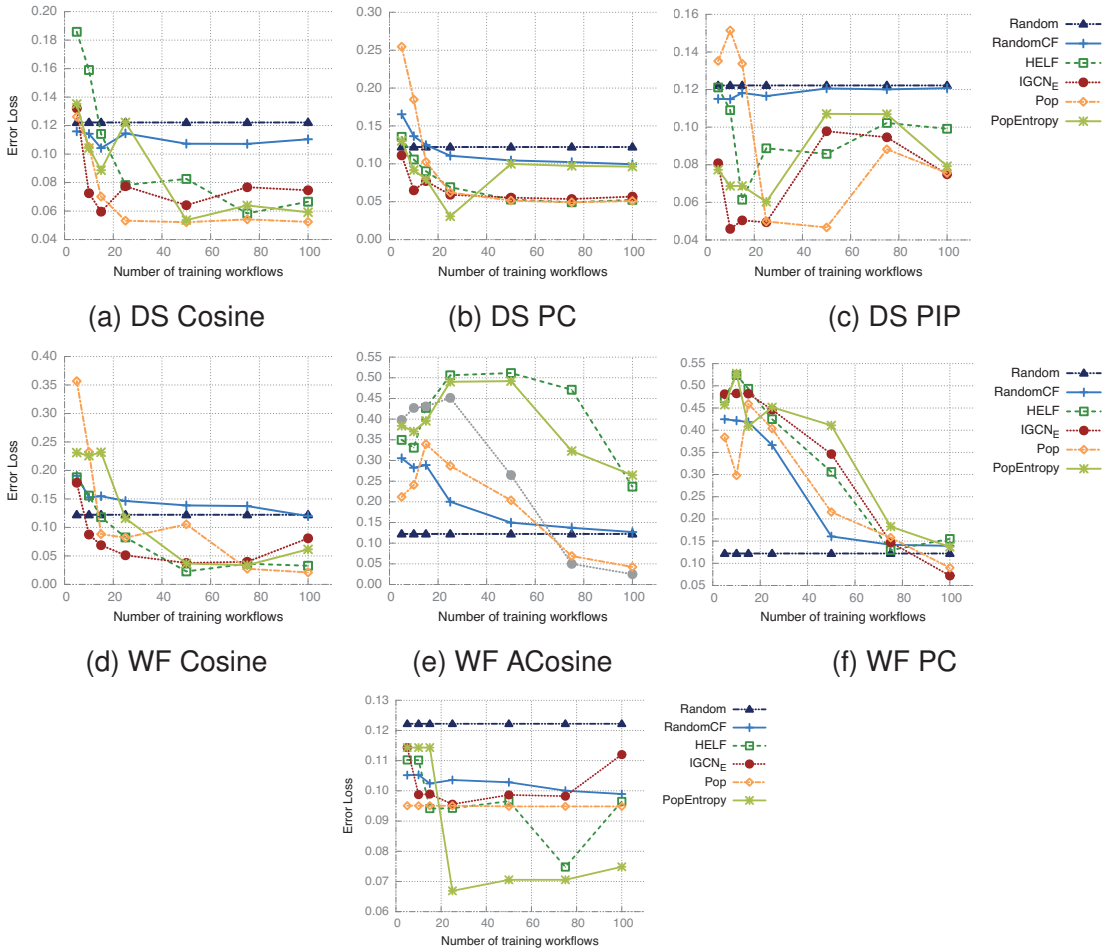


(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC

(g) WF SlopeOne

Figure 5.17.: NAR Loss between Top3 predicted and best real workflow for Dataset and Workflow-based CF



(a) DS Cosine (b) DS PC (c) DS PIP

(d) WF Cosine (e) WF ACosine (f) WF PC

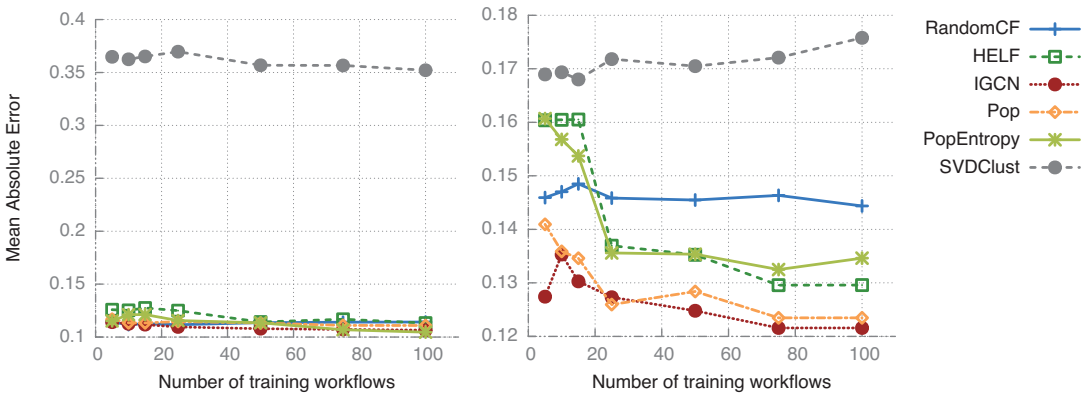
(g) WF SlopeOne

Figure 5.18.: NAR Loss between Top5 predicted and best real workflow for Dataset and Workflow-based CF

As the number of workflows in Top N increases, the random approach gets better. The same is true for the CF methods. The experiments show that classic CF works well also for regression tasks. However, the small number of datasets makes it not as successful as for classification problems.

5.5.2. Model-based CF

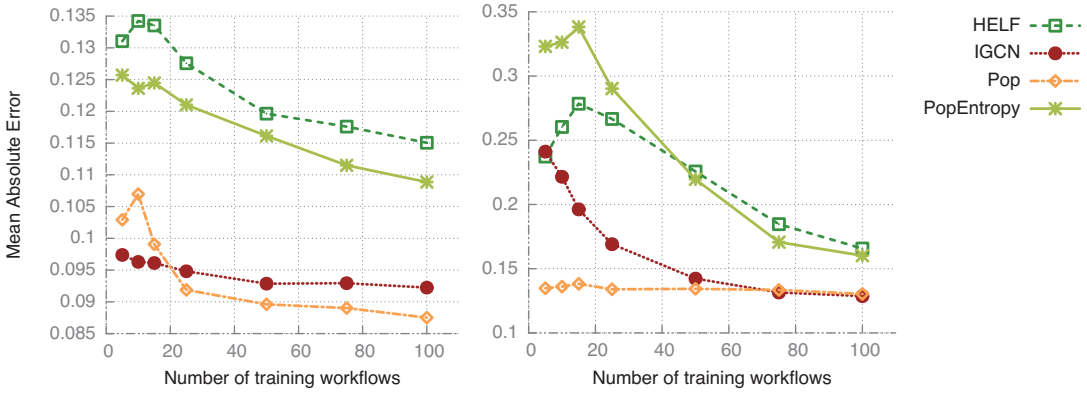
For model-based CF we have evaluated two clustering methods. First, we use a variant of k-means called k-medians where the absolute Euclidean distance is used as a distance measure and the median represents the centroid for each cluster. The second approach uses SVD to reduce the dimensionality of the matrix into a set of dataset features and workflow features. These features are used to cluster similar datasets. For a new dataset, workflows are selected from clusters based on their popularity and then the process is repeated to identify the cluster it belongs to. SVD for sparse matrices is used for clustering, however the results are not excellent. Figure



(a) Clustering classification (b) Clustering regression

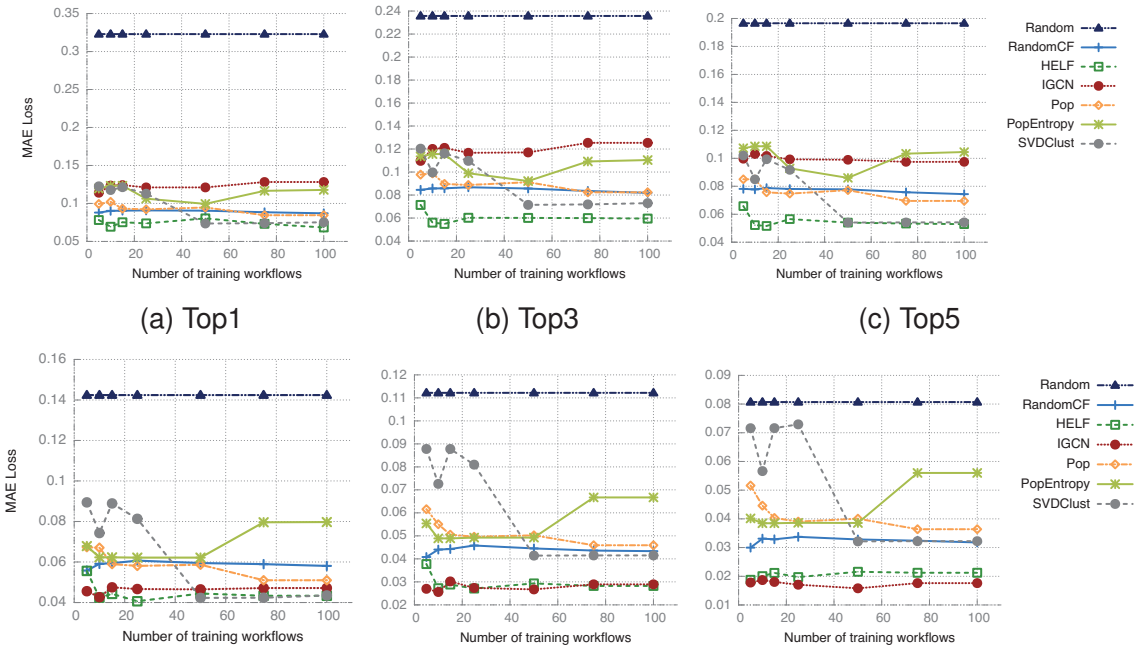
Figure 5.19.: MAE for Model-based CF (classification and regression tasks)

5.19 shows the results for both classification and regression problems in terms of MAE. As we can observe the MAE drops with the number of selected workflows for training. The performance of the k-Medians clustering is very good, but the one for SVDClustering is quite bad in terms of overall ratings. We present separately the results for the incremental SVD learning algorithm since we have selected the training workflows using the same four strategies as for the classic CF methods. The results show that the number of workflows used for training do not improve significantly the results. The results for MAE are shown in Figure 5.20.



(a) Classification (b) Regression

Figure 5.20.: MAE for incremental learning SVD (classification and regression tasks)

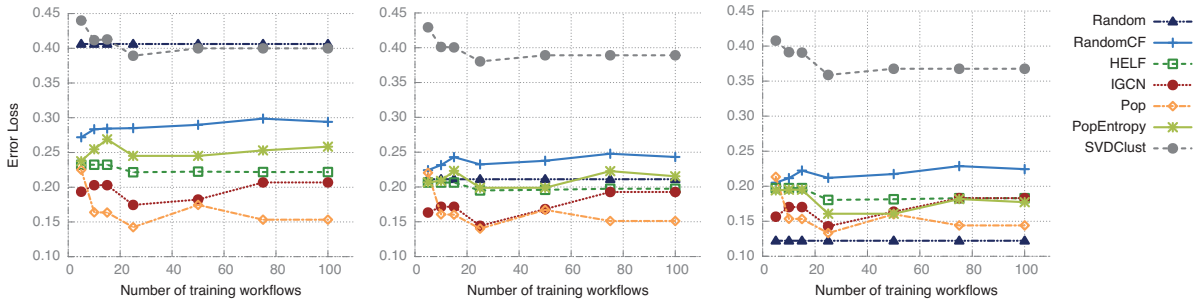


(a) Top1 (b) Top3 (c) Top5

(d) Top10 (e) Top15 (f) Top25

Figure 5.21.: MAE Loss between Top1-25 predicted and best real workflow for Model-based CF for classification

In terms of quality of recommendations the k-Medians approach is performing much better than the SVDClustering. The best performing method is k-Medians with HELF used for selecting the training workflows. The online selection also performs quite well. All the methods are better than the trivial random.



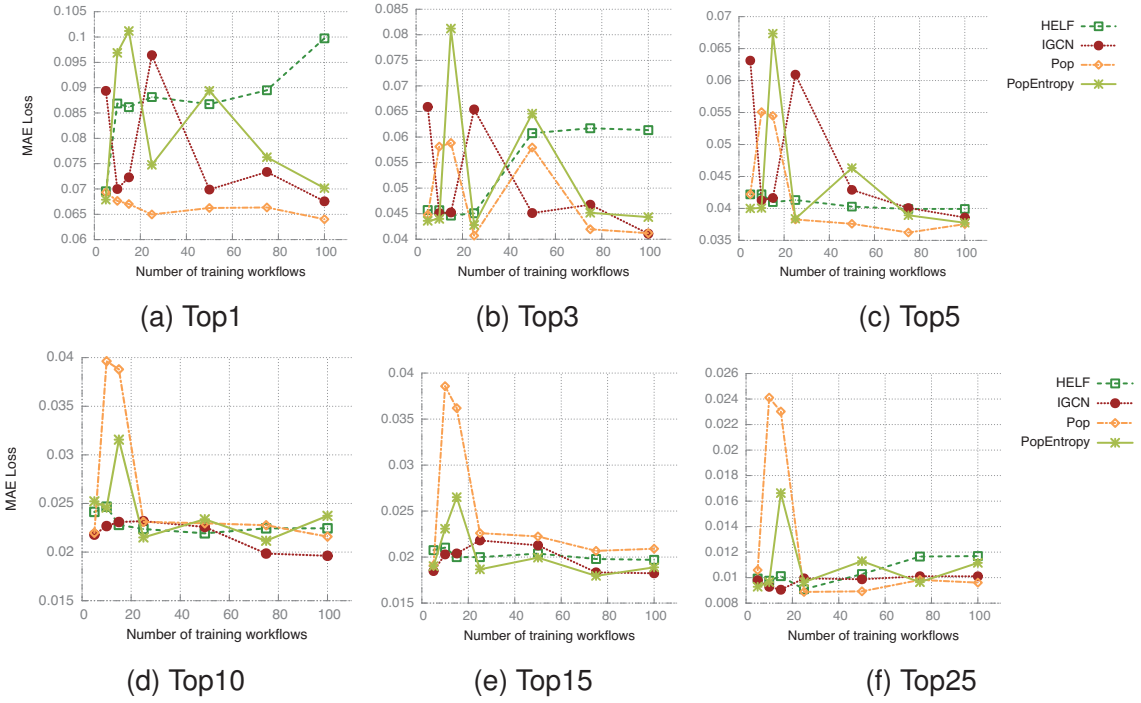
(a) Top1 (b) Top3 (c) Top5

Figure 5.22.: NAE Loss between Top1-5 predicted and best real workflow for Model-based CF for regression

As for the incremental learning SVD the results show that the loss is smaller than with previous methods. However, some of the methods oscillate from a small to a higher loss (IGCN, HELF, etc.). The loss starts from 0.7 and goes to 0.01. The values are smaller than before however the algorithm runtime is longer since it uses gradient descent with several steps. For regression problems the Pop method has the best results. The error loss decreases with the number of workflows used for training. The results for HELF and IGCN are quite modest.

5.5.3. Content-based CF

The content-based methods are basically relying on the datasets' features. For classification we use 12 features and for regression 8 features. We have eliminated similar features based on the correlation matrix. We tested k-NN, linear regression and regression tree (all available in Matlab). Two other methods are compared, first one using also k-NN but a different similarity measure (See Section 4.3.3) and the second one relies on the applicability of workflows (binary matrix). The results in terms of MAE for both classification and regression are presented in Figure 5.25. k-NN methods are the best performing ones for classification problems. The worst performing method is linear regression probably due to the reduced number of examples (at least for regression). Tree regression and the first k-NN method give similar results. Compared with previous methods the results are not as good: for both memory and



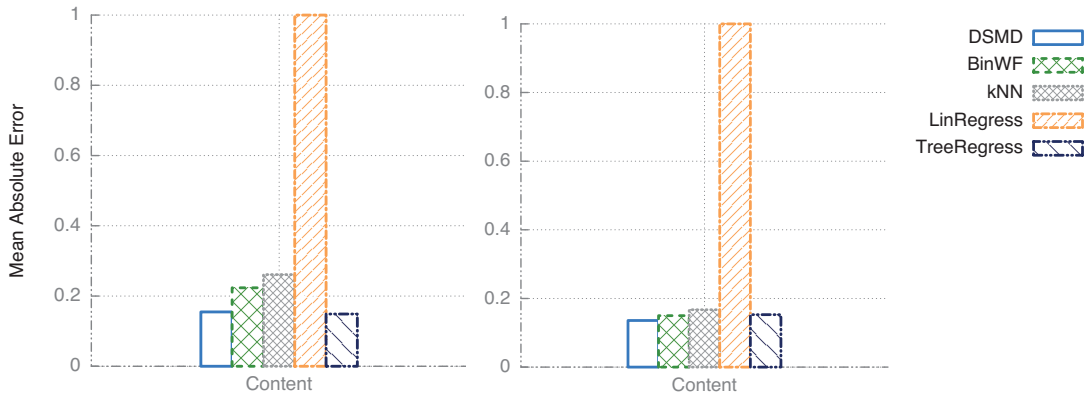
(a) Top1 (b) Top3 (c) Top5

(d) Top10 (e) Top15 (f) Top25

Figure 5.23.: MAE Loss between Top1-25 predicted and best real workflow for SVD-Inc for classification

(a) Top1 (b) Top3 (c) Top5

Figure 5.24.: NAE Loss between Top1-5 predicted and best real workflow for SVDInc for regression



(a) Clustering classification (b) Clustering regression

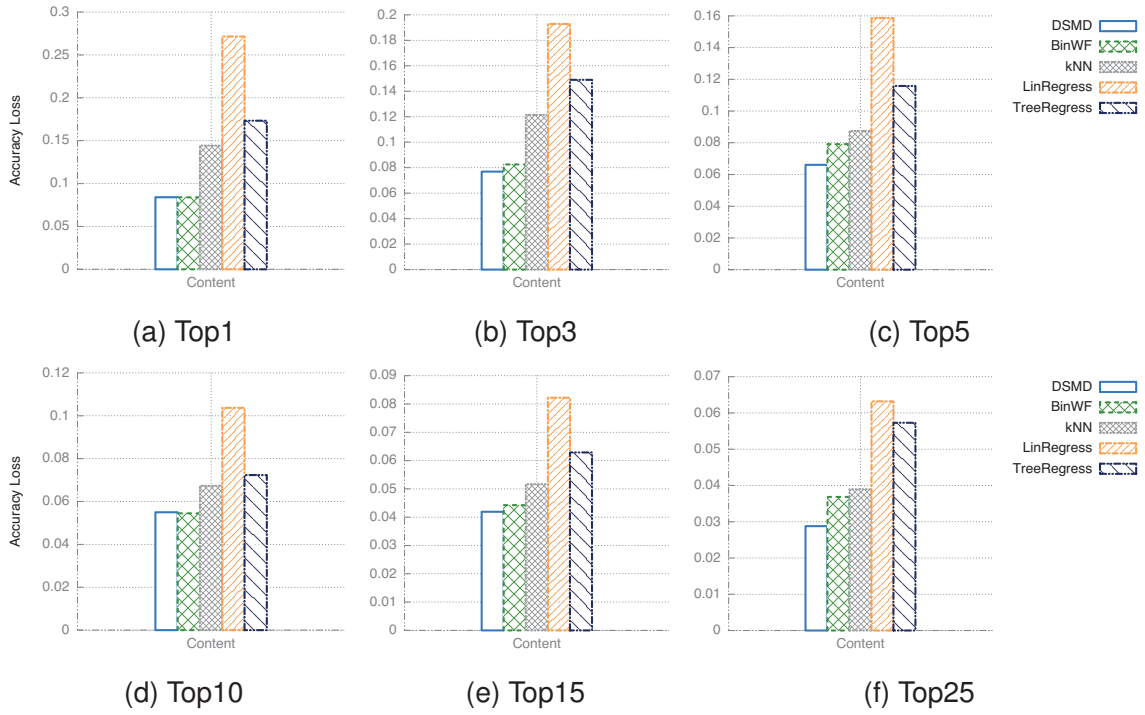
Figure 5.25.: MAE for Content-based CF (classification and regression tasks)

model-based CF the MAE is under 0.1 and for the content-based approach is almost double (0.2). However, for regression they give similar results because the case-base and the number of ratings are relatively small. Figure 5.27 shows the performance of the content-based CF methods in terms of quality of TOP N recommendations. When looking at the quality of best from TOP N recommendations for classification we can observe that the k-NN method and the binary one are leading. Compared with previous methods, they are a bit worse but still give good predictions. For regression problems the behavior is similar except for Top 1 where the Tree regression and k-NN (MATLAB implementation) are performing significantly better. Compared to other methods the results are slightly worse.

5.5.4. Hybrid-based CF

As discussed in Section 4.5 classic CF methods can be improved by combining them with content-based methods to provide more accurate predictions. We first tried to see if filling in the ratings for the target dataset would help in predicting the ratings. However, the results were not very good, the MAE increased significantly (around 0.22 instead of below 0.1). This is because the meta-data of the data does not characterise its performance on different workflows.

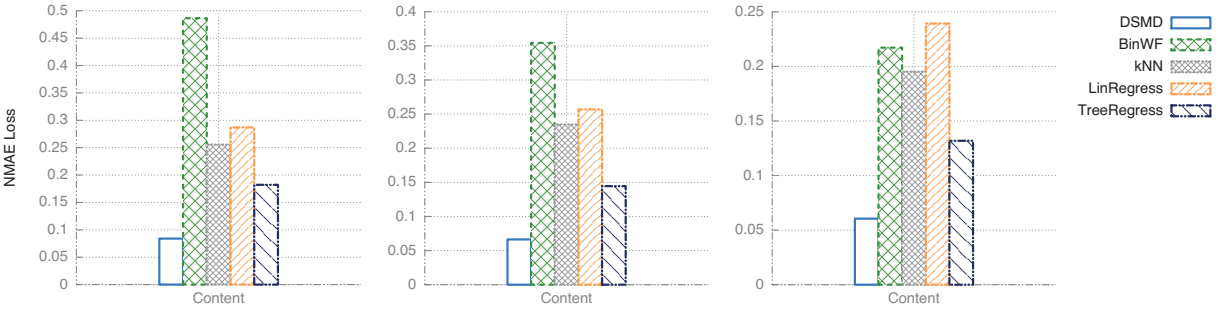
The second experiment took dataset-based CF (Cosine similarity) and the MD approach, computed the predictions separately and then averaged them. The outcome in terms of MAE was better than MD but worse than the normal CF methods (but for some methods a bit better in terms of regression error loss). Actually we can see an improvement if the number of training workflows is small and a loss when the number



(a) Top1 (b) Top3 (c) Top5

(d) Top10 (e) Top15 (f) Top25

Figure 5.26.: Accuracy Loss between best predicted workflow Top1-25 predicted and best real workflow for Content-based CF for classification



(a) Top1 (b) Top3 (c) Top5

Figure 5.27.: Error Loss between Top1-5 predicted and best real workflow for Content-based CF for regression

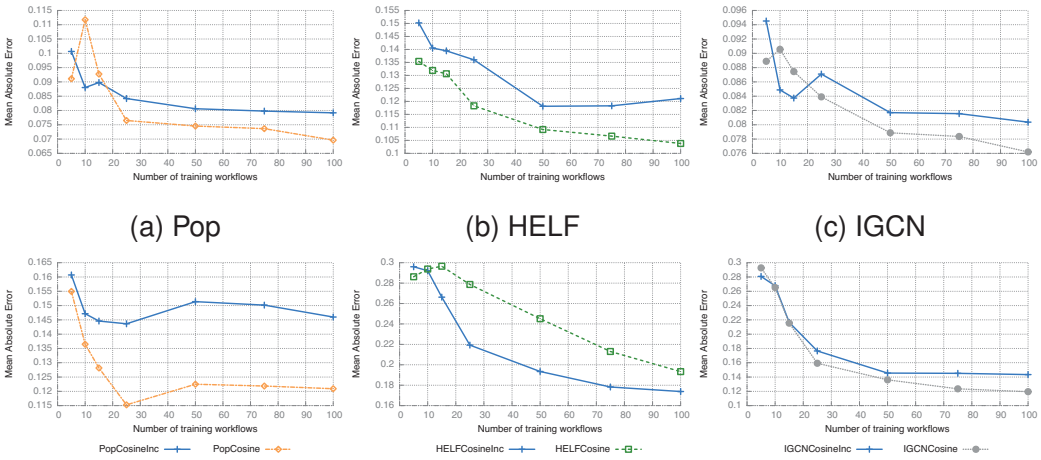
gets bigger (> 75). We can assume that CF and auto-experimentation improves the content-based approach but not always the other way around. The question is then if there is a combination of weights where the hybrid method would provide better results. We have varied the weights for CF and content approach and we observed that the influence of the CF approach has to be equal or higher than the one of the content-based approach in order to get an improvement. If the content approach gets a more significant weight then the results are getting worser than the CF approach.

5.5.5. Incremental Learning

All the previous presented methods are using a full matrix (zeros are only the non-applicable workflows). To test how the methods handle missing values in the matrix we devised an experiment that considers only the executed workflows for every tested dataset. Once being used for testing the dataset is added to the matrix and will contribute to the prediction of new datasets but this time with only a few ratings. We tested the Pop, IGCN and HELF strategies using Dataset-based CF with cosine similarity. For the model-based approach we tested the kMedians clustering.

The results for MAE can be seen in Fig. 5.28. As we can see the performance of the methods degrades smoothly for classification datasets. For Pop and IGCN the loss is only .01. Unexpectedly, the HELF measure improves its performance with .04 for regression workflows. For this approach, workflows have less ratings since there are more missing ratings leading to a change in the distribution of ratings (entropy as well). That means the workflows selected by HELF change as compared to the initial method. It may indicate that HELF may be biased towards workflows with many ratings that may not necessarily be the best indicators of representative workflows.

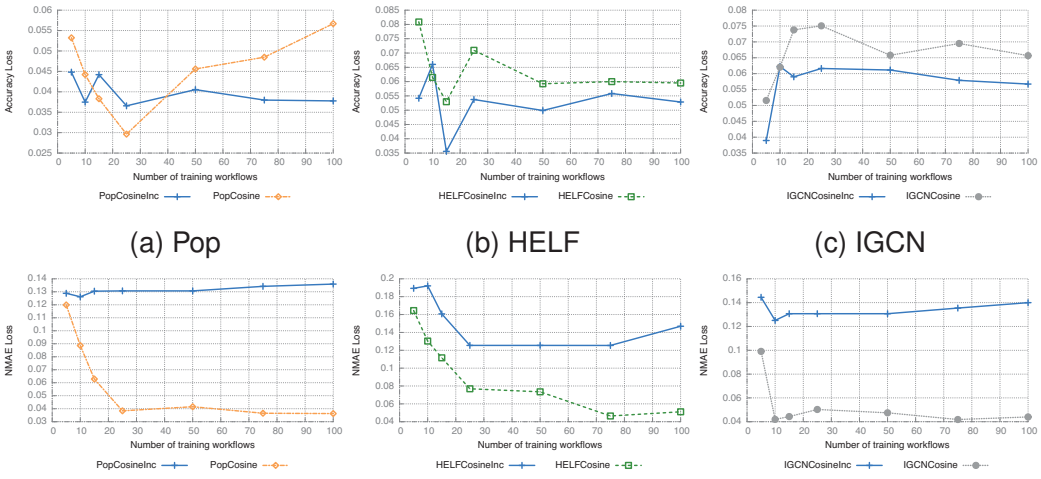
The regression problems degrade quite fast almost with 0.06. The loss is high compared to the results for classification. However, this can be expected due to two main reasons: First, the CF matrix for regression problems is relatively sparse (we have workflows that were rated by only 2-3 datasets). Second, the number of datasets is less than half than the one for classification problems. The results for Top K recommendations are shown in Figure 5.29. We consider only Top 10 for classification and Top 5 for regression problems. The results confirm the previous findings for MAE. The accuracy loss degrades with only .01 for Pop and IGCN, and increases with .02 for HELF. Again for regression problems the loss is more severe (between .06 and .07). We got similar results for model-based CF methods (kMedians) and for content-based (MD).



(a) Pop (b) HELF (c) IGCN

(d) Pop (e) HELF (f) IGCN

Figure 5.28.: MAE for incremental approach (classification a-c and regression d-f)



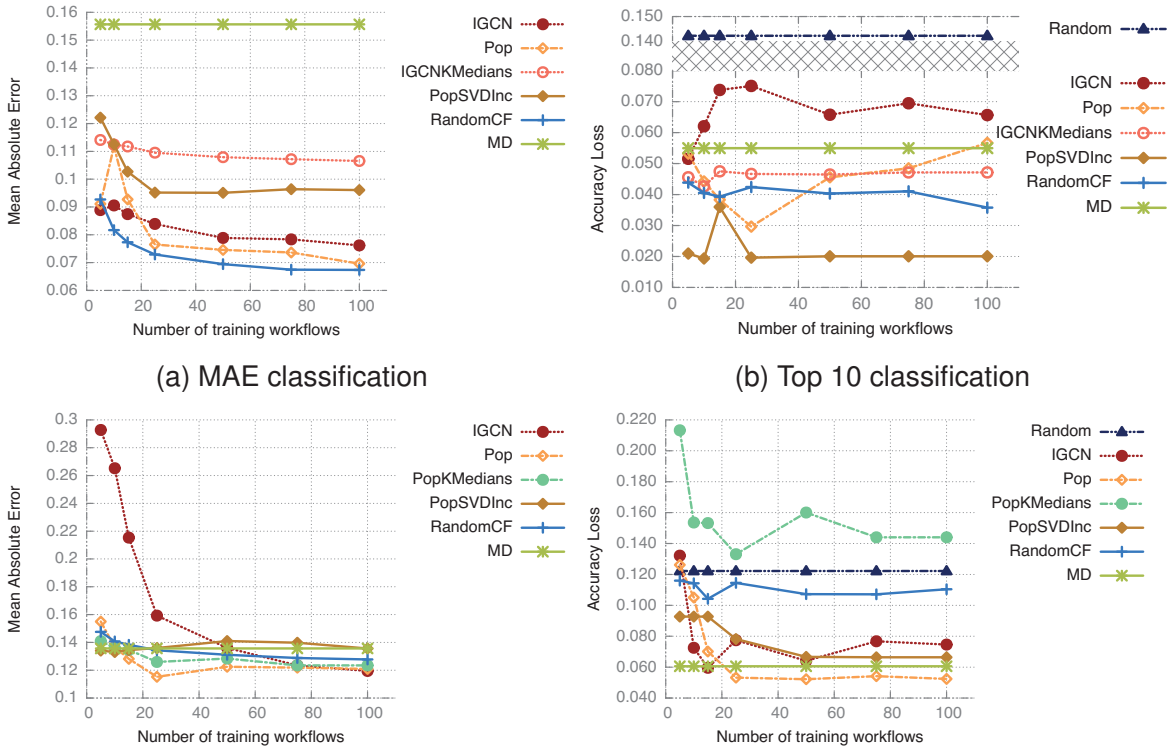
(a) Pop (b) HELF (c) IGCN

(d) Pop (e) HELF (f) IGCN

Figure 5.29.: Accuracy/Error loss for classification Top 10 and regression Top 5

5.6. Comparison of Methods

We have presented the performance of several CF methods combined with different selection strategies. In the following we are comparing the most successful methods from all categories. For classification problems we have chosen dataset k-NN with IGCN, Pop and Random as selection strategies, for model-based IGCNKMedians and PopIncSVD. For regression datasets we have dataset k-NN with Pop, HELF and Random as selection strategies, and PopKMedians and PopIncSVD. In addition, for both methods we employ the most successful content based-method MD. The CF methods use the cosine similarity (similar results are obtained for PC similarity). For the Top K measure we also compare with random selection. The results are presented in Figure 5.30. For classification problems, the best performing method for MAE is



(a) MAE classification

(b) Top 10 classification

(c) MAE regression

(d) Top5 regression

Figure 5.30.: Comparison of methods in terms of MAE and Accuracy/Error Loss (classification a-b and regression c-d)

dataset-based kNN with random selection, for accuracy loss it is outperformed by the PopSVDInc method. The results show that CF significantly outperforms the pure ran-

Method		5	10	15	25	50	75	100
IGCN	MAE	34	37	37	39	34	38	37
	Top 10	13	18	18	13	15	14	17
Pop	MAE	35	30	32	34	38	38	38
	Top 10	11	14	14	14	14	13	9
IGCNKMedians	MAE	31	33	32	31	32	33	33
	Top 10	14	15	15	14	14	13	13
PopSVDInc	MAE	26	27	27	32	32	32	32
	Top 10	18	19	18	19	19	19	19
RandomCF	MAE	29	34	36	37	41	42	41
	Top 10	19	20	22	24	24	22	23

Table 5.2.: Number of datasets for which the specific method outperforms MD for MAE (p = 0.01) and Top 10 (classification problems)

dom selection (for Top K) as well as the content-based method MD. We can observe that with 10 or 15 training workflows we get an error of 0.08 for MAE and $0.02 - 0.03$ for the accuracy loss for best workflow of Top 10. For regression the results are not as precise. The content-based method works quite well in terms of error loss. CF needs 25 training workflows for Pop to return a MAE of 0.115 and an error loss of ~ 0.05. This shows that CF is very sensitive to the number of ratings in the matrix.

We are interested to see for how many datasets the CF methods perform better than the MD method. For MAE we employ the Wilcoxon signed rank test for each dataset. For Top 10 we count the datasets for which the CF method performs better than MD. The results for classification problems (Table 5.2) show that the CF and model CF methods outperform the content-based method on more than 30 datasets (out of 44) for MAE. For Top 10 we get approximately 10-20 datasets for which the respective CF method performs better than MD (we did not count ties). To understand how each specific method performs on each dataset we show their boxplots in Figure 5.31. We can see three different types of behaviors: datasets for which IGCN and MD are both low (datasets 1, 6, 12, 25,) IGCN is better than MD (datasets 2, 3, 4, 5, 7, 8,

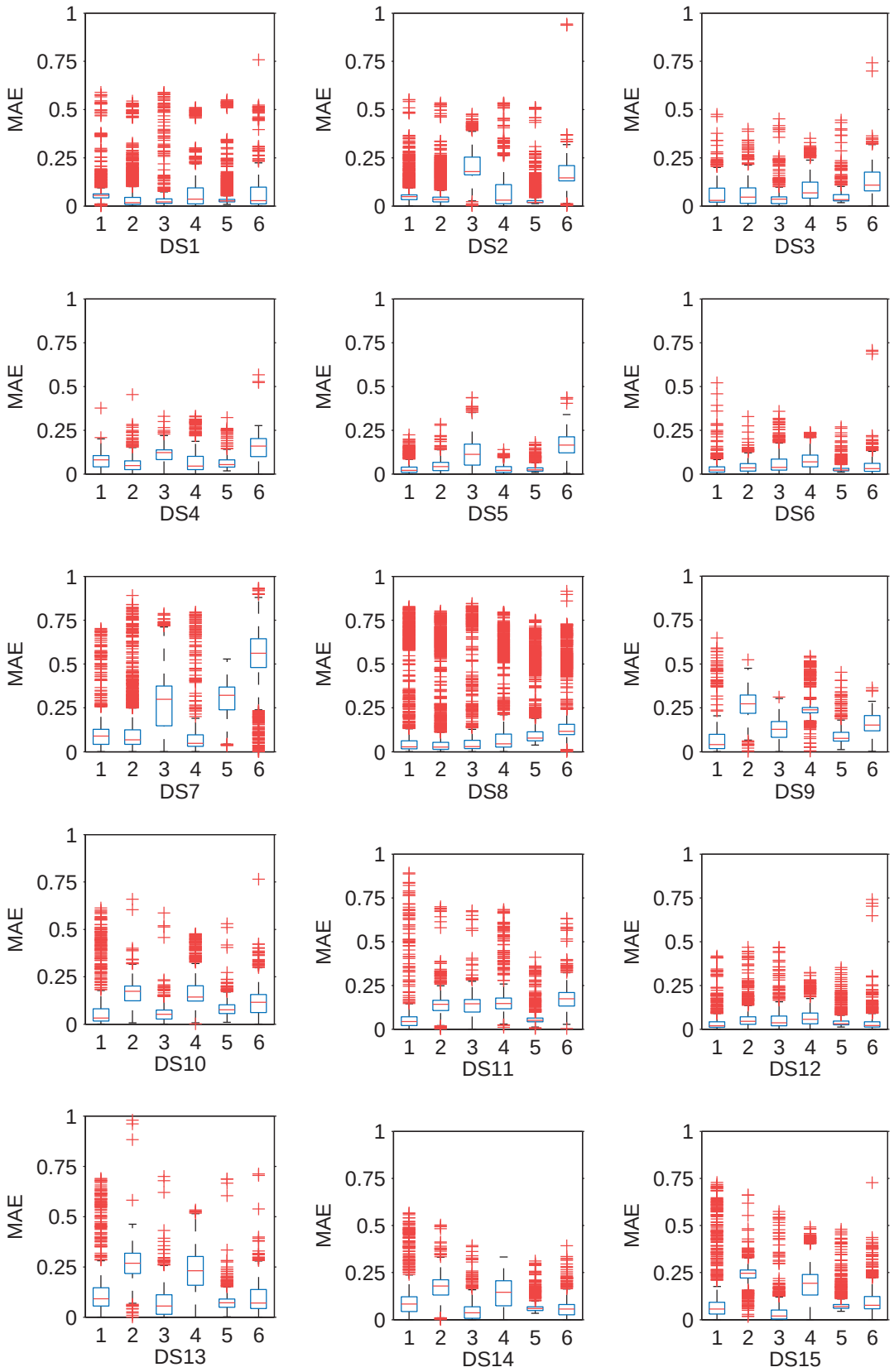
Method		5	10	15	25	50	75	100
IGCN	MAE	2	2	2	4	4	5	5
	Top 10	6	6	7	7	8	5	7
Pop	MAE	3	4	4	5	3	4	4
	Top 10	6	9	8	7	6	5	6
PopKMedians	MAE	7	6	6	8	8	8	8
	Top 10	5	7	7	4	4	4	4
PopSVDInc	MAE	4	3	4	3	3	3	3
	Top 10	5	4	4	5	5	5	5
RandomCF	MAE	9	2	2	2	3	3	3
	Top 10	4	2	5	5	6	6	6

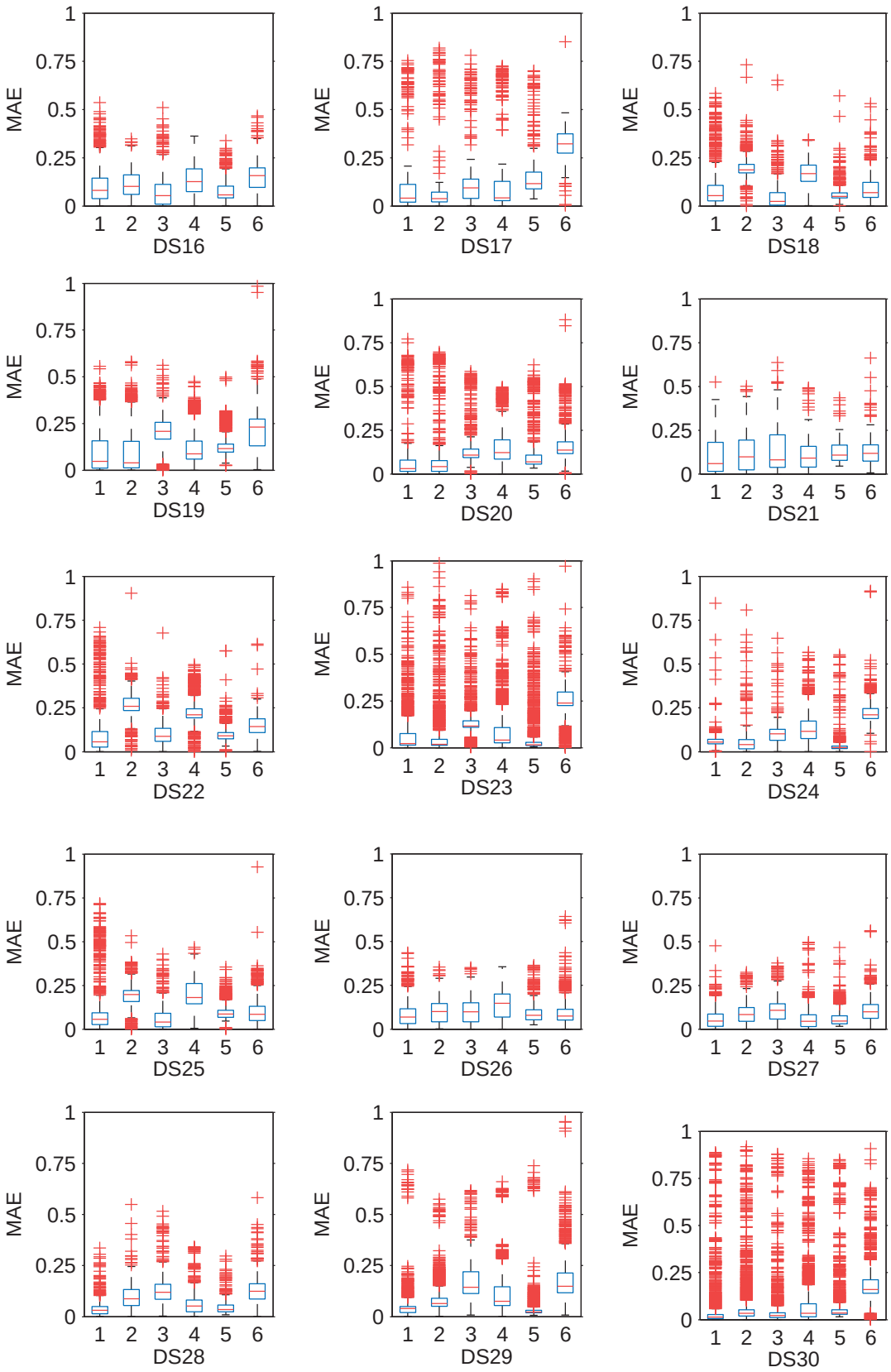
Table 5.3.: Number of datasets for which the specific method outperforms MD for MAE ($p = 0.05$) and Top 5 (regression problems)

9, 10, 11, 16, 17, 18, 19, 20, 22, 23, 24, etc.), MD is better (datasets 13, 14, 15), or both are not so good (datasets 21, 31). There are also some datasets for which we have many outliers (workflows whose rank cannot be predicted): 7, 8, 17, 20, 23, 30, 39 and 43.

We have also performed the Wilcoxon signed rank test for the methods used for regression problems. As expected we have less datasets for which the CF methods are better than the MD. The number of datasets for both MAE and Top 5 are shown in Table 5.3. Pop and PopKMedians are the best methods. For most of the remaining datasets MD and the corresponding CF method perform the same. In the case of regression the meta-features seem to be good descriptors of problems. However, also CF gives similar results. We think that the performance could be improved if more datasets were added to the CF matrix.

The boxplots for the 20 testing datasets used for regression can be seen in Figure 5.32. The number of training workflows used for CF is 25. These show the mean, standard deviation, min and max values in terms of MAE for each of the methods used in the comparison. Again we have 4 types of performance: Pop and MD perform


MAE
1
0.75
0.5
0.25
0
1 2 3 4 5 6
DS1
DS2
DS3
DS4
DS5
DS6
DS7
DS8
DS9
DS10
DS11
DS12
DS13
DS14
DS15


MAE
DS16
DS17
DS18
DS19
DS20
DS21
DS22
DS23
DS24
DS25
DS26
DS27
DS28
DS29
DS30

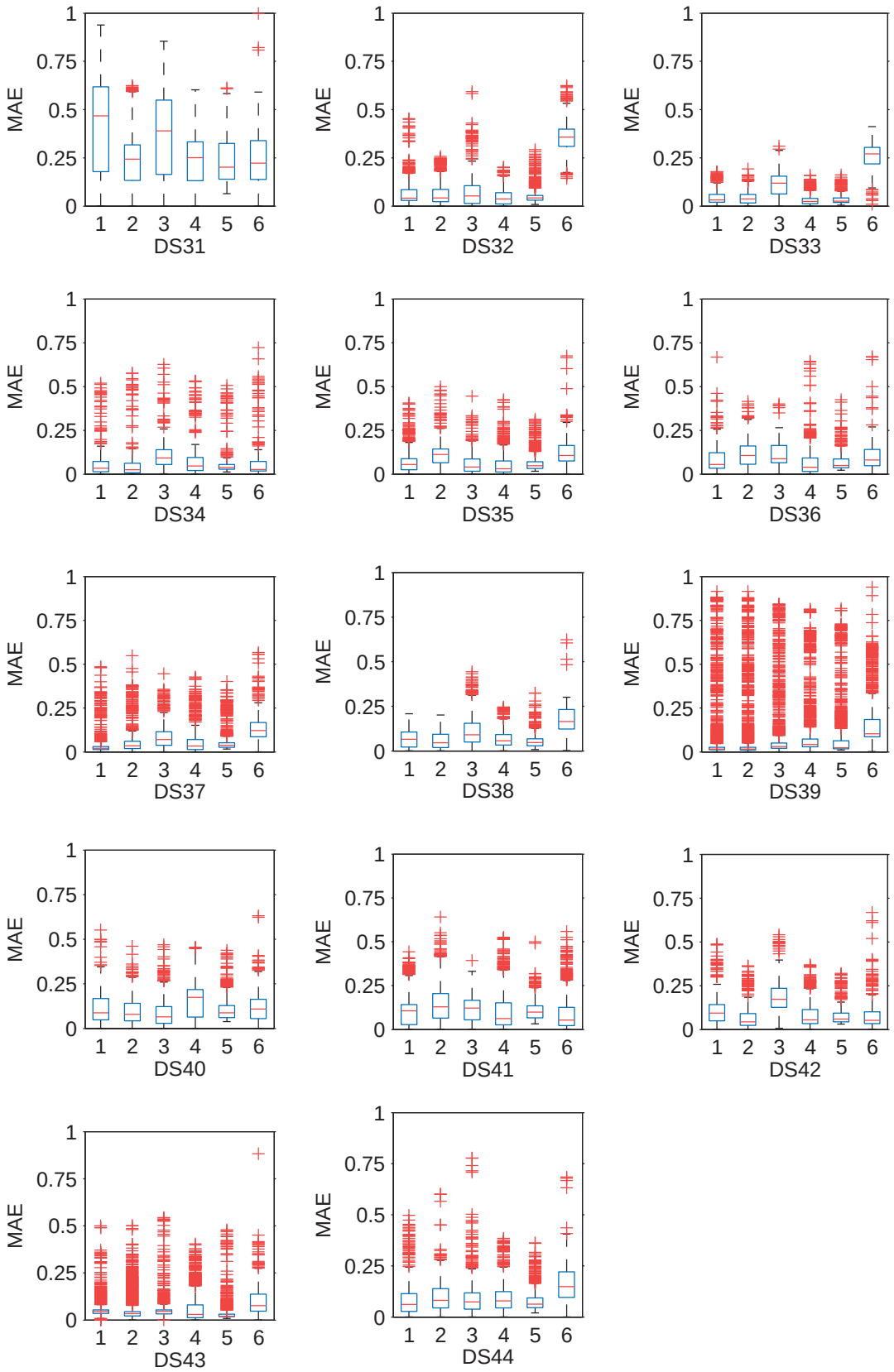


Figure 5.31.: Boxplot for testing datasets per methods (1 - IGCN, 2 - Pop, 3 - IGCNK-Medians, 4 - PopSVDInc, 5 - RandomCF, 6 - MD)

both well (datasets 1, 2, 5, 6, etc.), Pop is better than MD (datasets 9, 16), MD is better(datasets 7) and both are quite bad (datasets 3, 4, 14, 15, 17, 18).

5.7. Type of problems

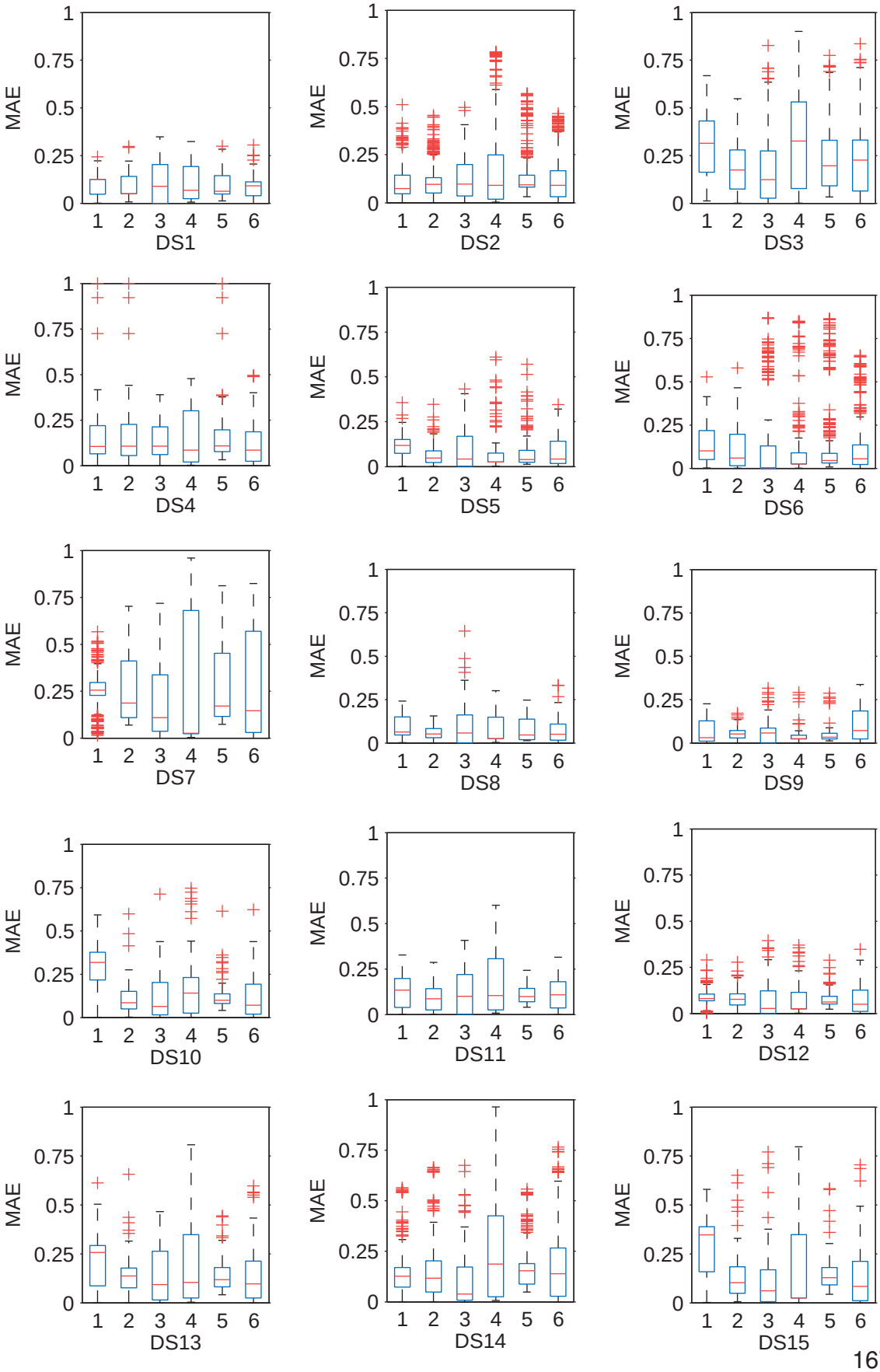
As presented in 5.1 datasets can be grouped by their distribution of ratings into the following categories: easy (many well-performing workflows), medium (an average number of well-performing workflows) and hard (few well-performing workflows). It would be interesting to find out how do the CF methods perform for each of these types and why are some better than others. This distribution is visible especially for classification problems. There are several easy datasets (2, 8, 9, 19, 23, 24, 29, 30, 39, 43). When looking at the performance of the RandomCF compared to other strategies, for many of these datasets the random selection performs slightly better (2, 23, 24, 29, 43). There are much more average or medium problems for which RandomCF and the other strategies perform relatively similar. As for the hard datasets (7, 9, 10, 11, 13, 14, 15, 16, 17, 18, 22, 32), there are a few datasets for which the performance of the methods is comparable (11, 13, 14, 15, 16, 18, 32) and some for which the offline and online strategies perform better (7, 9, 17, 22).

5.8. Limitations

The current approach relies on different types of resources that could be changed or improved: ML algorithm used, datasets, CF algorithms, etc.

These experiments were performed using a modified version of the ontology (from July 2011) to keep the consistency of experiments. Therefore, the generated workflows do not take into consideration the concept of column groups (attribute that share the same characteristics). This is relevant when we want to use operators that are applied only on specific columns. However, due to the fact that running all the workflows for many datasets is a very expensive process we have considered only operators that work on all columns at the same time (discretize all, fill in missing values all, etc.). This is one of the constraints that we have imposed. However, a new set of experiments can be performed in order to cover additional column-wise operators and to compare the behavior of the new operators and workflows.

We have considered datasets from different online repositories that cover both artificial and real-world problems. However, many of the datasets (e.g., from UCI) are preprocessed and alleviate the data analysis process. An idea for further testing


MAE
DS1
DS2
DS3
DS4
DS5
DS6
DS7
DS8
DS9
DS10
DS11
DS12
DS13
DS14
DS15

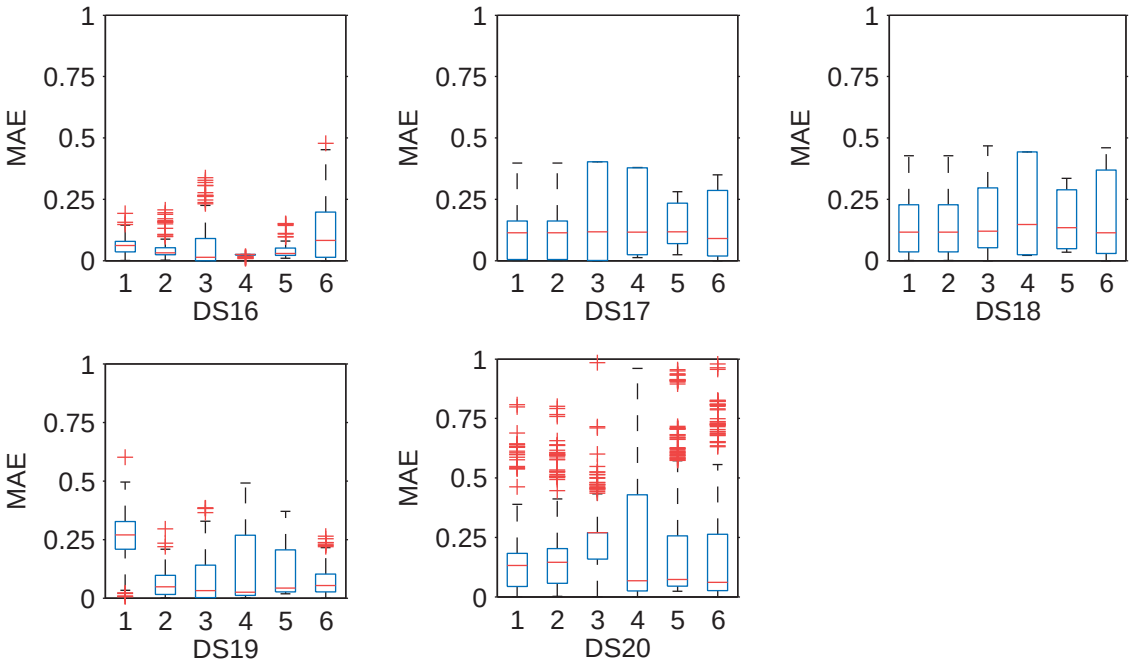


Figure 5.32.: Boxplot for testing datasets per methods (1 - HELF, 2 - Pop, 3 - PopK-Medians, 4 - PopSVDInc, 5 - RandomCF, 6 - MD)

and extension is to validate our approach on more datasets from real-world domains. This would allow verifying how important is the dataset's domain. One could also investigate if problems from similar domains have similar performance on the same workflows and if they could be clustered together.

We used the state of the art methods from CF field (kNN, clustering, SVD). Improvements could be obtained by using some methods that were proved to perform better than classic CF (e.g., pLSA, SVD with improved parameters, etc.).

The number of ratings in the CF matrix is another limitation of the current approach; especially the number of datasets compared to the number of workflows is quite small. The results for classification show that CF works better if the matrix is larger than for regression problems. The number of datasets for regression problems should be increased.

5.9. Discussion

We employed auto-experimentation and CF to solve the problem of ranking KDD-workflows. First, we have adapted the CF approach from movies$\times$items to datasets$\times$workflows.

We have showed how missing ratings for workflows can be computed in general. We have presented several classic CF methods in the context of ranking workflows. Some model-based approaches were introduced (clustering, SVD). Then features for datasets and workflows were extracted and used for content-based CF. Second, we have investigated the cold-start problem for new datasets. To solve this issue we have adapted some techniques from CF to our use-case. Third, we have considered the scenario when more workflows have missing ratings and therefore the sparsity in the CF matrix increases. We called this incremental learning and have explored what is the influence of sparsity on the presented CF methods.

Our experiments show the following: First, CF can be used to rank workflows with good precision. Second, for new datasets several selection strategies can be employed to improve the ratings over random selection. Two types of problems were analyzed: classification and regression tasks. For classification problems the results show that one can recommend Top 10 workflows and ensure that the best performing workflow is on average only .03 worst than the best one. For regression the results are still confirming that CF works, but the error in prediction is higher due to the small number of datasets used in our experiment. This allows us to accept the first two hypotheses (4.1 and 4.2) of Research Question 4.

Third, for classification problems the performance of CF methods degrades gracefully with the sparsity in the CF matrix. However, the performance for regression problems deteriorates much faster because of the small amount of ratings in the matrix. The results of the incremental learning approach confirm our last hypothesis that the CF ranking approach degrades gracefully in the presence of sparsity in the CF matrix (Hypothesis 4.3).

6

Role of Pre-processing

KDD is more than just applying patterns and finding models, but first the data needs to be prepared and transformed in a format that allows to further apply classic ML techniques. This is the task of pre-processing, which consists of several methods for analyzing raw data like data cleaning, data integration, data transformation, etc. In that context, pre-processing is a very important step of the KDD process. Some people even argue that when analyzing data one should spend more time on pre-processing and understanding the data than in all the other steps [Pyle, 1999, Kriegel et al., 2007, Zhang et al., 2003]. The methods employed in this step can influence considerably the performance of the overall KDD-process.

In this empirical study we analyze the results of our experiments (workflows) with focus on the pre-processing steps. As our HTN grammar allows both pre-processing and no pre-processing operators for the same workflows, we can compare the results. This allows us to detect workflows, datasets and situations when pre-processing improves or deteriorates the quality of the KDD process. The focus of the analysis is on three different pre-processing steps: filling missing values, discretization and normalization.

This chapter is organized as follows: First we shortly introduce the related work, then we present the results for filling missing values. This is followed by discretization and normalization. The last part discusses the findings and concludes the study.

6.1. Related Work

Different studies have shown that pre-processing impacts the outcome of data analysis. The work of [Crone et al., 2006] focuses on how different pre-processing techniques influence the performance of some classification algorithms (SVMs, DT, NN) for application domains such as operations research. They test several methods for sampling, scaling, coding of continuous and categorical attributes.

Several discretization methods have been discussed in the related work [Dougherty et al., 1995]. They compare supervised and unsupervised methods with specific focus in equal width interval binning, Holte's discretizer and a recursive entropy-based method. They report results for C4.5 and Naive Bayes on several UCI datasets and report improvements compared to when continuous attributes are kept. The entropy-based methods bring improvements to some of the algorithms. We also use bin discretization with different bin sizes, but we evaluate workflows not algorithms. In our experiments we do not evaluate any entropy-based discretization method.

Overall there are not many studies around about pre-processing and its impact with main focus on the evaluation of new methods or methods that fit better to specific types of data [Berka and Bruha, 1998].

In our study we look at workflows for which pre-processing provides statistically better results and also the ones for which pre-processing influences negatively the performance. We use the Wilcoxon signed rank test (one tail with p=0.05) to asses the importance of pre-processing as described in [Demšar, 2006]. For cross-validation one could use McNemar test which computes the wins, losses and ties per cross-validation. This is considered as a next test since it is more expensive to compute (compared to the number of workflows we need to compare).

6.2. Filling missing values

Data comes from real world scenarios where the data collection process is not ideal and many external factors can influence the process (unreliable sources, a failure in a device, etc.). This introduces missing fields in the data where the values are not known. When analyzing the data one must assess the impact of such missing values and try to either ignore, drop or fill in the missing values. In the ontology used for the experiments, we only considered either ignoring or filling missing values with different statistics. For the filling part we have used very simple methods like minimum, maximum, average, and zero. For our experiment we consider only those

Method Type	# workflows $=$	# workflows $>$	# workflows $<$
Maximum	127	61	4
Minimum	117	72	3
Zero	125	66	1
Average	129	58	6

Table 6.1.: Fill missing values vs. 'no fill' method for training workflows (p=0.05)

datasets and workflows that can allow both techniques [1]. In total we have 768 pairs of workflows where the first workflow keeps the missing values and the second one replaces them (4 different operators). The statistics that we have been looking at are: the number of workflows for which filling missing values performs equally/better/worse than normal workflows where missing values are kept. For each pair of workflows we applied a Wilcoxon signed rank test. Table 6.2 presents the number of pairs for which filling missing values works better and the ones for which it does not. We can observe that there are more successful workflows when fill missing values is applied rather than when it is not. The results are a bit different for the testing datasets. Here we test the accuracies after applying the model. Contrary to previous results the number of workflows when fill missing values works better is 0, most of the workflows performing equally well. Even if tested on the training data (with 10-fold cross-validation) the outcome is the same. The results are shown in Table 6.2.

All workflows that perform worse when filling the missing values are using 'DataToWeights' for weighting and 'NaiveBayes' as a data mining step on around 25-27 datasets (with two exceptions for 'WeightByInformationGain' and 'RMChaid' or 'kNNNominalMeasures'). For the cases when the fill missing values operator performs better the weighting method is 'DataToWeights' with 'kNNMixedMeasures' , 'kNNNominalMeasures', 'DecisionStump', 'CHAID', 'RandomTree', 'DecisionTree', 'NaiveBayes' and for most cases this is true for more than 20 datasets. For the datasets for which filling missing values is a bad idea we can see two patterns: first datasets with many missing values (proportion of missing values is high, > 0.05, e.g.AdultKDD, Ozone, Secom, CJS, etc) or datasets with very few missing values (< 0.08, Adult, ImageSegmentation, InternetAds, HallOfFame, etc.). We have expected to see a stronger impact of filling missing values. However, the main conclusion is that for many combinations of operators missing values do not impact the performance. Nev-

[1] There are some operators that do not work with missing values. We have excluded these from the study since we do not have their equivalent with missing values.

Method Type	# workflows $=$	# workflows $>$	# workflows $<$
Maximum	190	0	2
Minimum	192	0	0
Zero	192	0	0
Average	192	0	0

Table 6.2.: Fill missing values vs. 'no fill' method for testing workflows (p=0.05)

Method Type	# workflows $=$	# workflows $>$	# workflows $<$
Binning	1687	115-[15, 29, 32, 30, 9]	283-[40, 42, 52, 50, 99]
Frequency	1908	99-[18, 11, 8, 9, 53]	78-[31, 7, 7, 3, 30]
Size	1569	144-[55, 44, 30, 7, 8]	372-[28, 33, 43, 127, 141]

Table 6.3.: Discretize vs. 'no discretize' method for training workflows (p=0.05)

ertheless, there are some cases when it makes a difference and that is for datasets with a high and low number of missing values. For the first case it would have been interesting to test methods that drop the records with missing values.

6.3. Unsupervized Discretization

Discretization is a pre-processing method that allows to transform continuous attributes into categorical ones. There are several possible methods that can be applied. In our ontology we have included simple methods like 'Discretize by Binning', 'By Frequency' and 'By Size' (with 2, 3, 5, 7 and 10 bins) giving a total of 15 methods.

The total number of pairs of workflows is 6255 with 2085 workflows for each discretization method (and 417 for each bin type setting). We organize the results by category of binning. Datasets from the case-base produce the results in Table 6.3. The number of workflows for which results are statistically different is relatively low compared to the total number of workflows. The best performance is achieved when discretizing by frequency (more workflows are better than worse). As for the other two methods they have almost double the number of workflows for which they perform worse. The best results are obtained when the number of bins is set to 3, 5 and 7. However, when discretization by frequency is used 10 bins are much better. As for the test datasets the results can be seen in Table 6.4. The distri-

Method Type	# workflows =	# workflows >	# workflows <
Binning	1827	106-[10, 26, 34, 32, 4]	152-[39, 21, 34, 18, 40]
Frequency	1843	184-[25, 35, 38, 39, 47]	58-[24, 8, 8, 6, 12]
Size	1714	114-[43, 42, 22, 4, 3]	257-[20, 43, 48, 46, 100]

Table 6.4.: Discretize vs. 'no discretize' method for testing workflows (p=0.05)

bution of statistics is the same as for the case-base datasets. The workflows for which discretization produces bad results have the following characteristics: DM step – 'NaiveBayes', 'NaiveBayesKernel', 'RuleInduction', 'DecisionStump', 'RandomForest', 'RandomTree', 'SingleRuleInduction', and 'DecisionTree' with different combinations of weighting and filling missing values operators. As for the good performing ones: 'DataToWeights', 'WeightByRelief', 'WeightByUserSpecification', 'WeightByUncertainty', 'WeightByChiSquareStatistics', 'WeightByInformationGain' with 'NaiveBayes', 'NaiyeBayesKernel', 'RandomForest', 'DecisionStump', 'RandomTree', etc. There are not any specific datasets for which discretization works always better or always bad.

6.4. Normalization

Normalization consists of several methods that correct differences in the data by scaling it to fit a specific range. For normalization we consider only two methods: Z-transformation and range transformation. This brings a total of 4208 workflows, 2104 for each of the methods, but only for workflows that work with both normalized and un-normalized data.

A sound comparison is available in Table 6.5. The number of ties, wins and losses are presented in the columns. For case-base datasets the Z-transformation measure is performing better for more workflows than the range one. This is valid also for test datasets. The number of workflows for which it makes a difference to apply normalization is quite small. The Z-transformation workflows for test datasets which work better have the following features: DM step – 'QuadraticDiscriminantAnalysis' and 'RegularizedDiscriminantAnalysis' with different weighting operators, and the range one only for 'LinearDiscriminatAnalysis' with 'ChiSquaredStatistic'. The workflows that work bad for test datasets have the following most common features: 'NaiveBayesKernel', 'SVM_LibSVM_CSVC', 'AutoMLP'. The results change for the case-base datasets as follows: Bad workflows have as DM step for Z-transformation 'LinearDiscriminant-

Method Type	# workflows $=$	# workflows $>$	# workflows $<$
Z-transformation	1818	241	45
Range-transformation	1847	134	123
Z-transformation	1958	86	60
Range-transformation	2107	1	86

Table 6.5.: Normalize vs. 'no normalize' method for case-base respectively test datasets (p=0.05)

Analysis' (8 workflows), 'SVM_LibSVM_CSVC' (6 workflows), 'SVM_LibSVM_nuSVC' (6 workflows), 'AutoMLP' (5 workflows) and 'NaiveBayesKernel' (5 workflows), and for range transformation 'SVM_LibSVM_CSVC' (76 workflows), 'SVM_LibSVM_nuSVC' (14 workflows), and 'NaiveBayesKernel' (9 workflows). The good workflows consists of the following features: first for Z-transformation we have as DM step 'QuadraticDiscriminantAnalysis' (74 workflows), 'RegularizedDiscriminantAnalysis' (65 workflows), 'SVM_LibSVM_nuSVC' (16 workflows), 'RandomTree' (14 workflows), 'NaiveBayesKernel' (13 workflows), 'SVM_LibSVM_CSVC' (12 workflows), 'DecisionTree' (10 workflows), etc. Second for range transformation the DM step is one of the following: 'NaiveBayes' (27 workflows), 'QuadraticDiscriminatAnalysis' (25 workflows), 'RegularizedDiscriminantAnalysis' (16 workflows), 'RandomTree' (13 workflows), 'NaiveBayesKernel' (11 workflows), 'AutoMLP' (10 workflows), etc. The datasets do not differ very much for the good and bad workflows. We have looked at the kurtosis and skewness but we did not find any significant correlations.

6.5. Concluding Remarks

Pre-processing is an important step of the KDD process. This study has identified some situations when some of the pre-processing steps are influencing the performance of workflows. First, we have compared some missing values imputation methods with cases when the imputation is not done. We have confirmed that filling missing values is relevant when the data has a small number of missing values. We have also identified operators that are influenced by this step. However, other better performing operators should be tested in order to provide a more meaningful comparison. Second, we have looked into simple discretization methods and determined operators that perform well or bad when applying discretization. Third, we compared two normalization methods with workflows that do not apply normalization. We can

conclude that Z-transformation overall provides better results. Applying range transformation in most of the cases does not bring any improvement; on the contrary it may decrease the overall performance of the workflow.

Based on this study we can confirm that pre-processing is important, but this varies with the type of algorithms and characteristics of the datasets. Regarding the last hypothesis: we cannot reject the hypothesis since we have proof there are situations when pre-processing improves the performance of the KDD process. However, the number of workflows for which this happens is relatively small. For this reason we cannot accept the hypothesis but we can state that: 'For certain workflows pre-processing can influence the performance of KDD workflows.'

7

Limitations and Future Work

As with most novel ideas, our research about IDAs has several possible avenues for future development and improvement. We applied techniques from different fields to solve the problem of ranking and recommending KDD workflows. Each of the steps we followed in our research journey can be extended or optimised.

IDA survey The IDA survey is the result of a literature survey of papers relevant to the data analysis field with more focus on DM. By comparing several systems and analyzing their strengths and weaknesses we extracted a set of relevant features of IDAs. It would be interesting to conduct interviews with people from several fields and with different DM knowledge. The questions should focus on features they like or dislike in KKD tools. As well they could suggest new features or extensions. The user study would help to identify the desirable features for IDAs with a better precision.

DMWF Ontology The number of operators limits the DMWF ontology. At the moment only RM and some Weka operators are modeled in the ontology. Weka operators were not properly tested and should be better described in terms of their conditions and effects. Other operators from the RM plug-ins could be easily added, but one needs to understand how they work and provide correct conditions and effects. Also the parameters settings of operators could be explored to find out which

parameter values are good in what situation.

Another limitation of this work is the fact that it focuses solely on classification and regression tasks. To further improve the HTN more tasks and methods could be added depending on the DM task that needs to be solved. For example, one could provide HTNs for text or image mining. The modeling of operators is done manually, but one could learn conditions and effects for operators. This could be implemented as a RapidMiner plug-in and integrated with the existing IDA plug-in. Moreover, some of the features of eProPlan (e.g., testing of conditions and effects) could be moved to RapidMiner to provide a better integration. However, this would limit the usage to a single KDD tool.

A possible extension of this work is the development of DMWF ontologies for other KDD tools (e.g., KNIME, Orange, etc.). This would allow other tools to benefit from the planning and ranking capabilities of our plugin.

eProPlan To improve eProPlan's capabilities one could change the plan view to explore and better visualize the generated plans. At the moment eProPlan was tested only on planning DM workflows and on a small classic planning problem (missionaries and cannibals). One could build compilers for converting PDDL to DMWF-OWL (and the other way around). This would allow using eProPlan for the planning domain (e.g., planning web-service composition or planning classical problems) and compare the Flora2-planner to other planners. Another idea worth trying is replacing the planner with JSHOP (or other planner) and try to adapt the system. The planner is implemented in Flora-2 and needs different binaries for every operating system. JSHOP is Java-based and therefore platform-independent.

For auto-experimentation we have used a reduced number of operators since running all the generated workflows is very expensive. One could extend the ontology and run experiments with the Weka operators. Additionally, the column-based operators for pre-processing could be tested and analyzed.

CF for ranking KDD workflows The KDD workflow recommendations were made using straightforward CF methods. A limitation is the size and the structure of the CF matrix. Here missing ratings are non-applicable workflows. This could be changed by treating non-applicable workflows as the worst performing workflow – they get the worst ranking value for the corresponding dataset in the matrix. This would lead to a full rating matrix and therefore the cold-start problem would be solved. Other methods could be tested and adapted especially the ranking approaches (CofiRank [Weimer

et al., 2007]). We believe this could provide better recommendations since it has been shown that it performs better than classic CF methods.

We limited ourselves to CF, but there exist other ranking methods that it would be worth exploring. For example, one could use reinforcement learning to find the best performing algorithms for a new problem at hand. The cumulative reward could assess a workflow's performance on datasets using exploration of the state/action space. It would be interesting to see if that leads to similar results as the existing method.

An important factor in workflow recommendation is also the time needed to execute a specific workflow. Users may prefer workflows that offer a good trade-off between accuracy and time. A mixed metric considers precisely this combination: one of the measures described above and the execution time. Here, the ranks of workflows take into consideration both measures. Different weights could be used to compute the desired influence of each of the measures. Another interesting combination to follow is to combine several of the measures above with the execution time. However, we did not explore this metric in our present experiments. This remains as future work that can be easily tested since all the execution details are stored in the database.

Another limitation is the fact that we did not explore recommendations for the new item problem. New workflows could be generated for datasets depending on their meta-data and also on the ontology. One would need to execute the new items to create a case-base and then later use the ratings for CF. Similarly to the new dataset problem the existing approach could be extended to handle this issue.

For content-based CF we used the same weight for all the features. The interesting question is how to learn the best distribution of weights such that it minimizes the accuracy loss (e.g., look into genetic search for learning weights, etc.). At the moment all 12 features have the same weight and it may be the case that some of them are more important than others. We are interested in finding the weight distribution that leads to a more precise similarity. But how do we define when we are better? One could use the CF predictions: if the weights lead to a lower accuracy loss then the weight distribution is better.

We used incremental learning to test the proposed approach when the matrix is more sparse. As further work, we propose to add as training workflows also the ones recommended to the user (Top K). For each target dataset that was used for testing, use the training workflows but additionally also the Top 10 workflows that were recommended. This would increase the number of workflows for each testing dataset and probably improve the recommendations.

Pre-processing The study could be enhanced by performing additional experiments for other methods: fill missing values with prediction, removing attributes with too many missing values, entropy-based discretization, explore feature selection methods, etc.

The current approach is very generic and was tested on standard datasets from UCI and other ML repositories. However, datasets are very domain-dependent. As future work one could include domain knowledge into the recommendation/ranking system. Moreover, it would be interesting to determine how other real-world datasets perform with the current ranking system.

8 Conclusions

The KDD domain is growing as more and more algorithms are produced every year. KDD tools provide a large gamma of methods but the user support is rather limited. The users need in-depth knowledge and understanding of the operators to build good workflows. However, this requires time and experience and that is even difficult for professional data miners.

To counteract these limitations, this thesis has proposed a new approach: We combine ontologies and planning to automate the KDD process. The end product is a tool that produces correct workflows based on the characteristics of the dataset and the task description. This simplifies the data analysis process and excludes bad workflows.

We have taken the classic IDA one step further towards true support for data analysis: we recommend plans that are almost as good as the best ones. The thesis follows closely a set of research questions that we have pursued and for which we have found more explanations.

The first research question RQ1 tries to identify the problems of existing IDAs.

RQ1 *Why is KDD difficult for users and what are the main issues of today's tools?*

To investigate the above question we defined the following hypothesis:

HYPOTHESIS 1.1 *Current KDD tools have a limited support for users.*

To prove the hypothesis we have performed an extensive survey of the last 30 years of research from data analysis. The survey underlined the fact that current KDD systems have limited support for users, lacking features as automatic generation of workflows, workflow ranking, support for multiple steps, etc. The short lifespan of some of the approaches shows that most of the tools are only a proof of concept and cannot be used to solve real-world problems (see Section 2.2). This study helped us to identify important features that should be included or strengthened such that KDD tools improve their user support.

To solve one of the most critical problems of today's KDD tools—automatic workflow generation–we developed a system, eProPlan, that is able to automatically generate all the possible correct workflows for a given dataset and DM task. For this problem we have defined the next challenge or research question RQ2:

RQ2 *What characteristics should a KDD workflow ontology have to allow automatic generation of workflows?*

To solve this problem we defined one hypothesis:

HYPOTHESIS 2.1 *Ontologies and planning can be used to generate automatically all the correct workflows.*

Data Mining is only one step from the KDD process. Analyzing data is more than just applying the DM step: it requires understanding the data and going through all stages of the KDD process. Our approach follows closely the steps described in KDD cook-books or standards. We use the ontology language OWL-2 to describe the KDD domain for planning. As a first step, we found mappings for each element of AI planning (planning domain, start problem, end goal, operators, goals, methods, etc.) and described them into an ontological language. It lead to an extension of the SWRL language allowing to express all possible actions for planning. As ML algorithms or KDD operators have different parameters we had to find a way to model their behavior. We made use of classic constructs from ontologies (classes, properties, etc.). Inputs/outputs of operators are complex structures like tables. Operators are very sensitive to how the data looks like and for this reason we had to model the IOObjects at a very fine level of detail in terms of attributes/columns.

As more researchers looked into the automatic generation problem we have compared our work with existing approaches. We have found their work only scratches the surface and is mostly a proof of concept. They can not be used for real-world problems and were also not tested with real users. No other system could model KDD problems at such level of detail (column operations, parameters of operators,

all steps). Our ontology with focus on planning and workflows comprises more than 100 operators. Moreover, our approach has been integrated into on of the most used KDD tools, RapidMiner, and has been used by several people.

A challenge of the KDD domain is the fact that it is continuously expanding with new techniques. Every year new algorithms are developed and integrated in KDD tools. For example, RapidMiner allows users to easily develop new plug-ins. To solve this problem we designed the next research question:

RQ3 *How could the ontology be maintained and modeled over time?*

We analyzed and discussed the following hypothesis related to RQ3:

HYPOTHESIS 3.1 *The ontology needs to be updatable such that new algorithms can be added.*

KDD is growing rapidly and more techniques appear. This requires to adapt the ontology over time and extend it with new operators. Our plug-in allows to model, test and debug it. Special views support the modeling of operators' conditions and effects as well as defining new tasks and methods to simplify the planning process. The eProplan system is designed as a plug-in of the ontology tool Protégé. It is configurable, extensible and modular, consisting of several small modules that support this process.

However, being able to generate all the possible workflows for a given dataset and task does not simplify the problem for users. Imagine that users would get for a classification problem more than 1000 possible and correct workflows. The main challenge for them would be to pick a set of workflows to execute, but they may select only workflows that give bad results. This leads to our next research question:

RQ4 *How could one rank KDD workflows and recommend Top K workflows?*

We analyzed and discussed the following hypothesis related to Research Question 4:

HYPOTHESIS 4.1 *Collaborative filtering and Auto-experimentation can be used to rank the generated workflows.*

Our system was extended with a module that combines CF and auto-experimentation to provide rankings. Our experiments show that CF can be used to recommend workflows for new datasets. The results confirm the second hypothesis suggesting a good workflow in the Top 10 that only differs in accuracy from the real best workflow by 0.02

(see Section 5.5). Another experiment was devised to show how the CF ranking approach handles sparsity in the rating matrix. The results confirm the third hypothesis by showing a small drop in performance when the incremental method is used (see Section 5.5.5).

Having executed so many workflows for a large number of datasets we thought of the possibility of studying the effect of pre-processing step in the KDD process. For this purpose we followed the last research question:

RQ5 *How important is pre-processing in the KDD workflow?*

To investigate this problem we defined the following hypothesis:

HYPOTHESIS 5.1 *Pre-processing is improving the performance of workflows.*

The analysis of our experiments shows that pre-processing cannot be neglected from the KDD process. Filling missing values, discretization and normalization improve the performance of certain workflows (see Chapter 6). Since the results did not show the improvement at a large scale (for a large number of workflows) we did not accept the last hypothesis. However, we cannot reject it since there is evidence of workflows that improve when pre-processing is present in the KDD workflow.

Overall the work in this thesis showed that KDD tools need to be improved by offering a better user support and experience. This can be done by combining different research fields: AI planning, auto-experimentation, collaborative filtering and ranking. This leads to a new perspective on assistance for data mining that is able to automatically generate forkflows and recommendations for new problems. This simplifies the process for both novices and Data Mining experts.

Appendices

A
Tools

For this thesis several tools were developed. The overall system consists of several modules: ontological planning module, experimentation module, ranking module and user interaction module as shown in A.1. eProPlan is the first module that is responsible for generating all the correct workflows starting from a dataset and a task description. The experimentation-module is responsible for generating the case-base or the CF matrix. The CF module is using CF techniques to provide ratings for new datasets. Since it provides recommendations for KDD wokrflows we will call it for simplicity Algorithm Recommender System (ARS). The user interaction module is actually the RM-IDA where users can specify their wishes in terms of what data they want to analyse and what is the end goal. For the moment the new rankings are not integrated in the RM-IDA. That remains as a next step for development.

A.1. eProPlan

eProPlan was designed as tool that facilitates the modelling, testing and maintainability of the DM ontology. The system represents a Protégé plugin that consists of several tabs: The operator tab allows to define new operators with their conditions and effects. It has at its core an editor that checks the correctness and syntax of the declared rules. The Goal tab is basically used for retrieving the meta-data of new

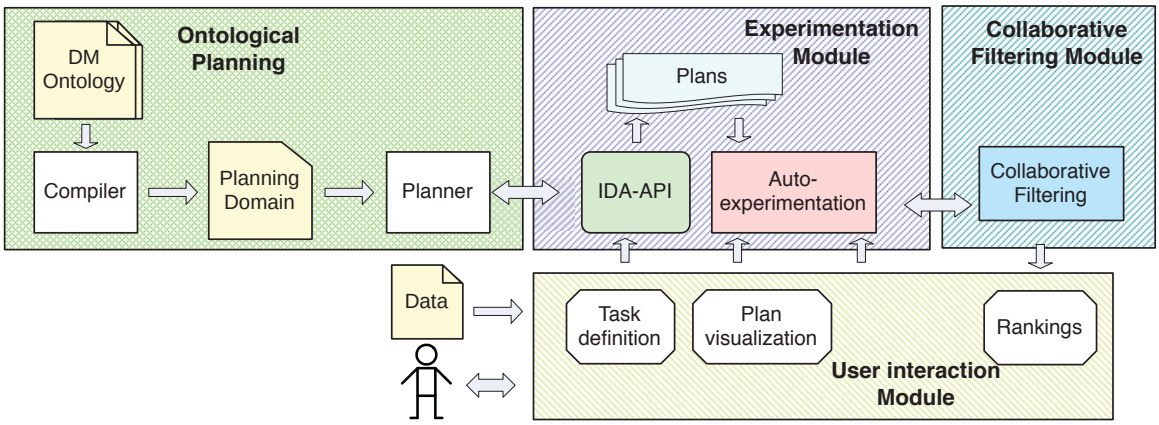


Figure A.1.: Overall System

datasets as individuals in the ontology. It also allows to declare instance of goals that need to be used for planning. The Plan tab is responsible for planning and checking every step of the planning process. It also displays the instances created and every step and the workflow itself. The Task and Methods tab is specialised in tasks and methods and the hierarchy itself. New hierarchies can be created to guide the planning process. The Builtins tab allows to define new language constructs that may be needed to express more specialised conditions and effects. Some of this tabs can be seen in the Figure A.2, and A.3. The eProPlan system was developed in Java,

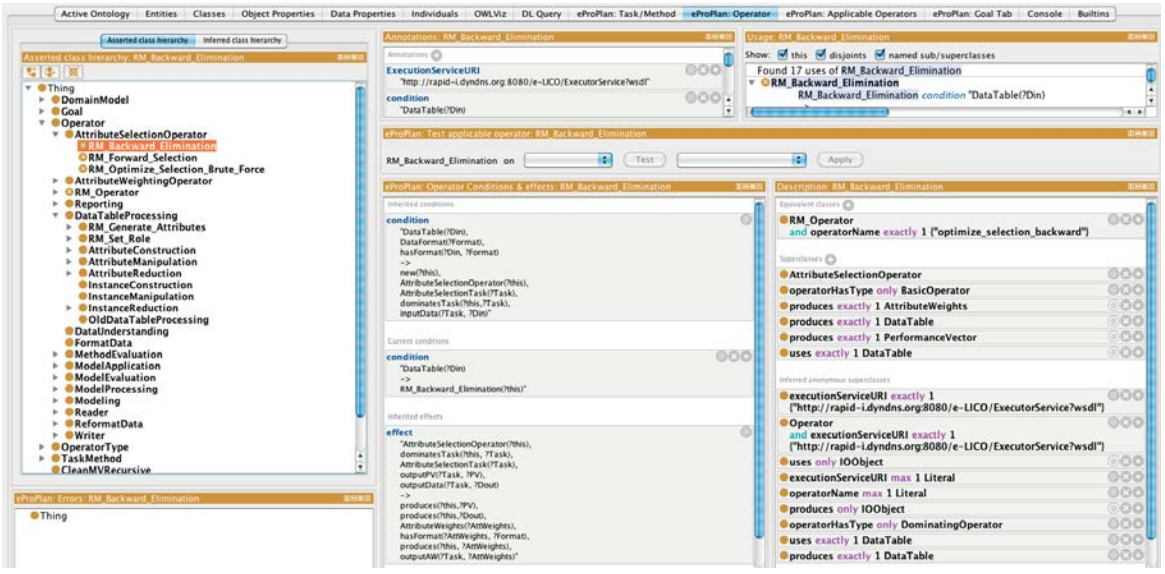


Figure A.2.: The Operator tab

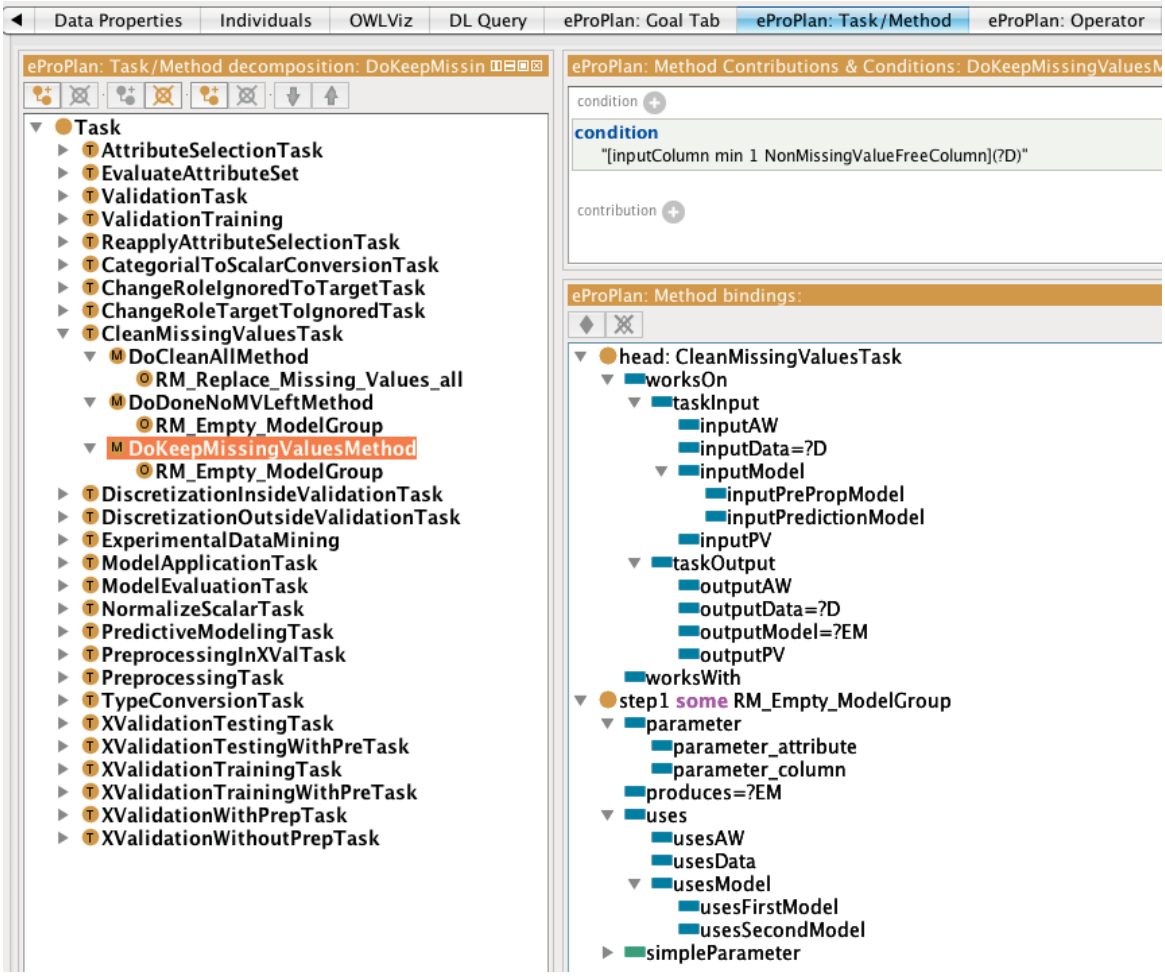


Figure A.3.: The Task&Method tab

the planner in Flora2 and XSB. The Interprolog [1] library was used to integrate the two languages.

A.2. ARS

This is a separate module developed in MATLAB. It relies on the CF matrix generated via experimentation. It contains several similarity measures for datasets and workflows, it makes predictions of missing ratings based on different strategies. We also include several selection strategies for the cold-start problem when new datasets need rankings.

A.3. RM-IDA

This plugin is based on the IDA-API and the planner. It includes an installer that automatically downloads all the required libraries according to the OS. The user only needs to have write rights on the installation folder. If the resources are updated then the user gets update message to download the latest version.

The plugin has several tabs and views that allow the user to analyze their data with only a few clicks. The user needs to have knowledge about the purpose of the analysis and also understand his/her dataset. Once the workflows are generated the user can select one and check its operators. Then the workflow can be executed to find how good it performs.

[1] http://www.declarativa.com/interprolog/

B
Datasets

B.1. Classification datasets

In total we have used 117 classification datasets from which 73 are in the case-base that are used only for training and the rest 44 were used for both training and testing. The datasets from the case-base can be seen in Table B.1 and the testing ones in Table B.2.

B.2. Regression datasets

We did not find as many datasets as for classification problems. The number of regression datasets is only 47, from which 27 represent the case-base and 20 are used for testing. Table B.3 contains the list with the regression datasets in the case-base and Table B.4 with the testing ones.

Id	Dataset name	Id	Dataset name	Id	Dataset name
1	Aids	26	Ecoli	51	PlanningRelax
2	Annealing	27	Eucalyptus	52	PostOperative
3	Arrhythmia	28	fl2000	53	PrimaIndianDiabetes
4	Asbestos	29	Flags	54	PrimaryTumor
5	AustralianCreditApproval	30	germangss	55	prnn_crabs
6	Authorship	31	Glass	56	ProfB
7	BackAche	32	Haberman	57	ScaleBalance
8	Baloons	33	Hayes_Roth	58	Schizo
9	Biomed	34	HeartC	59	Seeds
10	BrazilTourism	35	HeartDisease	60	Sonar
11	BreastCancer	36	HeartH	61	Soybean
12	BreastCancerWDiagnostic	37	Hepatitis	62	SpectfHeart
13	BreastTissue	38	HorseColic_SL	63	StatlogHeart
14	BroadwayMult	39	IndianLiver	64	StatlogVehicle
15	Cars	40	Ionosphere	65	TeachingAssistant
16	Cloud	41	Iris	66	TicTacToe
17	Collins	42	Irish	67	Trains
18	CreditApproval	43	Lenses	68	VertebralColumnC2
19	CylinderBands	44	LungCancer	69	VertebralColumnC3
20	cyyoung8092	45	Lupus	70	Vote
21	cyyoung9302	46	Lympho	71	Vowel
22	Dermatology	47	Mammographic	72	Wine
23	DMFT	48	MolecularBiologyPromoters	73	Zoo
24	Donated	49	Parkinsons		
25	Echocardiogram	50	PittsburghBridges_Mat		

Table B.1.: Training classification datasets

Id	Dataset name	Id	Dataset name	Id	Dataset name
1	Adult	16	MFeatMorphological	31	CJS
2	AdultKDD	17	MFeatPixel	32	Marketing
3	Car	18	MFeatZernike	33	EyeMovements
4	ChessKRKPA7	19	Mushroom	34	Connect4
5	Contraceptive	20	Musk2	35	RingNorm
6	GermanCreditApproval	21	Nursery	36	TwoNorm
7	ImageSegmentation	22	OptDigits	37	PokerHand
8	InternetAds	23	Ozone	38	Madelon
9	Isolet	24	PageBlocks	39	Secom
10	JVowel	25	PenDigits	40	SensorReadings24
11	LetterRecognition	26	Spambase	41	SteelPlatesFaults
12	MagicTelescope	27	WaveformV2	42	MolecularBiologySplice
13	MFeatFactors	28	Yeast	43	BankMarketing
14	MFeatFourier	29	Hypothyroid	44	fbiswc
15	MFeatKarhunen	30	HallOfFame		

Table B.2.: Testing classification datasets

Id	Dataset name	Id	Dataset name	Id	Dataset name
1	Automobile	10	Triazines	19	gsssexsurvey
2	AutoMPG	11	BodyFat	20	negotiation
3	ComputerHardware	12	CloudStatlib	21	female_bladder
4	ConcreteSlump	13	FishCatch	22	female_lung
5	Housing	14	FruitFly	23	male_bladder
6	Metadata	15	Auto93	24	male_lung
7	Servo	16	Lowbwt	25	boston
8	PBC	17	Sleep	26	pbco
9	Pharynx	18	Strike	27	sensory

Table B.3.: Training regression datasets

Id	Dataset name	Id	Dataset name	Id	Dataset name
1	Abalone	8	WineQualityRed	15	Bank8fm
2	CardiotocographyABC	9	WineQualityWhite	16	Quake
3	CardiotocographyNSP	10	2DPlanes	17	SolarFlareC
4	CommunityCrime	11	Ailerons	18	SolarFlareM
5	ConcreteData	12	Bank32nh	19	houses
6	ParkinsonsTeleMotor	13	CPU_Act	20	pbcseq
7	ParkinsonsTeleTotal	14	MV		

Table B.4.: Testing regression datasets

C
Evaluation

We present in detail some of the results that may be useful to understand how the methods perform.

C.1. Workflow statistics

We shortly present the overall performance of workflows. We focus on workflows that perform constantly well and workflows which perform very bad. We analyze both the case-base (workflows only used for training) and the testing workflows separately. We are especially looking at operator combinations that have on average very good or bad performance. We consider only the number of datasets for which the performance of workflows is distributed uniformly (sometimes workflows behave the same).

Table C.1 presents the workflows whose average is close to the best one for datasets in the case-base (for classification around 73 datasets). We can observe there is a set of DM algorithms that work the best and outperform the other (e.g., RuleInduction, NaiveBayes, NaiveBayesKernel, kNNMixedMeasures, DecisionTree, AutoMLP, ID3, NeuralNet). A similar trend can be seen for the weighting operator where DataToWeights and UserSpecification are most frequent. For discretization Size and Binning are more common with 1,10 bins respectively 3,5 bins. For normalization both

C. Evaluation

# Ds	Workflow description
38	RuleInduction + NormalizeZ/Range/NoNorm +WeightByDataToWeights/UserSpec.
	NaiveBayesKernel+WeightByUserSpec/DataToWeights
	kNNMixedMeasures+NormalizeZ+WeightByUserSpec/DataToWeights
	NaiveBayes + DiscretizeBySize (bins=7,10) + WeightByDataToWeights/UserSpec
	NaiveBayesKernel + DiscretizeBySize (bins=7,10) + WeightByDataToWeights/UserSpec
37	kNNMixedMeasures+NormalizeByRange+WeightByUserSpec/DataToWeights
	NaiveBayes+DiscretizeBySize(bins=10)+WeightByDataToWeights/UserSpec
	NaiveBayesKernel+DiscretizeBySize(bins=10)+WeightByDataToWeights/UserSpec
	DecisionTree+DiscretizeByFrequency (bins=3) +WeightByDataToWeights/UserSpec
	RuleInduction+DiscretizeBySize(bins=10)+WeightyByDataToWeights/UserSpec
31	RuleInduction+DiscretizeByFrequency(bins=3)+WeightByDataToWeights/UserSpec
	ID3+DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
	DecisionTree+DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
	NaiveBayes+DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
	NaiveBayesKernel+DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
29	NaiveBayesKernel+ReplaceMissingValues (zero) + NominalToNumerical+NormalizeByRange + WeigthByUserSpec/ValueAverage/DataToWeights
	AutoMLP+ReplaceMissingValues (zero) +NominalToNumerical+NormalizeZ +DataToWeights/UserSpec/ValueAverage
	NaiveBayesKernel+ReplaceMissingValues (minimum)+NominalToNumerical+NormalizeRange + WeightByDataToWeights/UserSpec/ValueAverage
27	RuleInduction+NominalToNumerical+NormalizeRange/NoNorm/NormalizeZ+WeightByDataToWeights/UserSpec
25	LinearDiscriminantAnalysis+ReplaceMissingValues (minimum)+NominalToNumerical+NormalizeRange+ WeightByDataToWeights/UserSpec/WeightByValueAverage
	RuleInduction+DiscretizeByFrequency(bins=5)+WeightsByDataToWeights/UserSpec
	LinearDiscriminantAnalysis+ReplaceMissingValues (average)+NominalToNumerical+DataToWeights/UserSpec
24	NaiveBayes+ReplaceMissingValues (minimum)+DiscretizeByBinning(bins=10,7)+WeightByDataToWeights/ValueAverage/UserSpec
	NaiveBayesKernel+ReplaceMissingValues(zero)+DiscretizeByBinning(bins=10,7)+WeightByUserSpec/DataToWeights
22	RuleInduction+WeightByValueAverage
	RuleInduction+NormalizeRange/NormalizeZ+WeightByValueAverage
	NaiveBayesKernel+NoNorm/NormalizeByRange+WeightByValueAverage
	kNNMixedMeasures+NormalizeRange+WeightByValueAverage
21	NeuralNet+NoNorm/NormalizeRange/NormalizeZ+WeightByDataToWeights/ValueAverage/UserSpec
	AutoMLP+NormalizeZ/NormalizeRange+WeightByDataToWeights/ValueAverage/UserSpec
20	RuleInduction+NormalizeZ/NormalizeRange+WeightByDataToWeights/UserSpec
	QuadraticDiscriminantAnalysis+ReplaceMissingValues (minimum)+NominalToNUmerical+NormalizeZ+ WeightByChiSquared/WeightByUnceratinty/WeightBySVM
19	RuleInduction+NormalizeZ+WeightByDataToWeights/UserSpec
	ID3+DiscretizeByFrequency(bins=7)+WeightByDataToWeights/WeightByUserSpec
	NaiveBayes+DiscretizeByFrequency (bins=7)+WeightByDataToWeights/WeightByUserSpec
	NaiveBayesKernel+DiscretizeByFrequency (bins=7)+WeightByUserSpec/DataToWeights

# Ds	Workflow description
38	SingleRuleInductionSingleAttribute + DiscretizeBySize (bins=3) + WeightByChiSquared
	kNNNominalMeasures+DiscretizeByBinning (bins=2) + WeightByRelief
	kNNMixedMeasures+DiscretizeByBinning (bins=2) + WeightByRelief
31	DefaultModel + DiscretizeByBinning (bins=2,5,7) + WeightByDataToWeights
29	kNNNumericalMeasures+ReplaceMissingValues (minimum,maximum)+NominalToNumerical+NormalizeRange+WeightByDeviation
	kNNMixedMeasures+ReplaceMissingValues (minimum,maximum)+NominalToNUmerical+NormalizeRange+WeightByDeviation
27	DefaultModel + NominalToNumerical+NormalizeRange+WeightByUserSpec
25	DefaultModel + DiscretizeByFrequency (bins=5) + WeightByChiSquared/ByUncertainty/ByRule
24	RegularizedDiscriminantAnalysis+ReplaceMissingValues (minimum)+ NominalToNumerical+WeightByDataToWeights
22	QuadraticDiscriminantAnalysis + WeightByDeviation/ByInfoGain
	RegularizedDiscriminantAnalysis + WeightByDeviation/ByInfoGain
21	RegularizedDiscriminantAnalysis+NoNormalize/NormalizeRange+WeightByValueAverage/UserSpec/DataToWeights
20	QuadraticDiscriminatAnalysis+WeightByUncertainty/WeightByRule
	QuadraticDiscriminatAnalysis+NormalizeRange+WeightByUncertainty
19	RegularizedDiscriminantAnalysis+NominalToNumerical+WeightByDataToWeights/ValueAverage/UserSpec
	DefaultModel+DiscretizedByFrequency (bins=7) + WeightByChiSquared/ByUncertainty

Table C.2.: Worst classification workflows from the case-base on average

Z-transformation and range are equally present; in some situations no normalization provides better or equal results.

The worst performing workflows can be seen in Table C.2. They have as DM step the following algorithms: kNNNominalMeasures, kNNMixedMeasures, DefaultModel, QuadraticDiscriminatAnalysis and RegularizedDiscriminantAnalysis. This are correlated with DiscretizeBySize or Binning (bins=2,5,7) and different weighting algorithms.

For the testing datasets the numbers are similar only that we have less datasets. As shown in Table C.3 the DM algorithms that perform well are almost the same as for the case-base. Therefore, the algorithms have a consistent performance. Table C.4 introduces the worst performing workflows for the testing datasets. The combination of kNNNominalMeasures, kNNNumericalMeasures and kNNMixedMeasures with discretization (both Binning or Frequency with bins=2,3,5) as well as weighting by uncertainty, using information gain or ratio, or chi squared provide represent the workflows with the lowest average accuracy.

# Ds	Workflow description
26	kNNMixedMeasures+NormalizeZ/ByRange + WeightByUserSpec/DataToWeights
	NaiveBayes/Kernel+DiscretizeBySize (bins=10,7,5) + WeightByDataToWeights/UserSpec
25	RuleInduction + NormalizeByRange/Z + WeightsDataToWeights/UserSpec
	RuleInduction + DiscretizeBySize/Binning(bins=10,7,5)+WeightByDataToWeights/UserSpec
	DecisionTree + DiscretizeByFrequency (bins=2) + WeightByUserSpec/DataToWeights
24	DecisionTree+DiscretizeBySize (bins=3,5,7,10) + WeightByDataToWeights/UserSpec
	RuleInduction + DiscretizeByBinning (bins=7,5,10) + WeightByDataToWeights/UserSpec
	RandomForest + DiscretizeByBinning (bins=7) + WeightByUserSpec/DataToWeights
21	AutoMLP + NoNorm/NormalizeByZ/ByRange + WeightByUserSpec/DataToWeights/ValueAverage
	kNNNumericalMeasures + NormalizeByZ/ByRange + WeightByUserSpec/ValueAverage/DataToWeights
	kNNMixedMeasures + NormalizeByZ/ByRange + WeightByUserSpec/ValueAverage/DataToWeights
20	ID3 + DiscretizeBySize/Binning (bins=3,5,7,10) + WeightByDataToWeights/UserSpec
	CHAID + DiscretizeByFrequency (bins=2) + WeightByUserSpec
19	NeuralNet + NormalizeByRange/Z/NoNorm + WeightByValueAverage/UserSpec/DataToWeights
	ID3 + DiscretizeByBinning (bins=3,5) + WeightByDataToWeights/UserSpec
17	kNNMixedMeasures + DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
	kNNNominalMeasures + DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
	NaiveBayes/Kernel + DiscretizeByFrequency (bins=3) + WeightByDataToWeights/UserSpec
16	SVMLibSVM_nu_SVC_Classification + NormalizeByZ/Range + WeightByValueAverage/DataToWeights/UserSpec
	DecisionTree + DiscretizeByFrequency (bins = 3) + WeightByDataToWeights/UserSpec
14	NaiveBayes/Kernel + DiscretizeByFrequency (bins=5) + WeightByDataToWeights/UserSpec
	kNNMixedMeasures + DiscretizeByFrequency (bins=5) + WeightByDataToWeights/UserSpec
	kNNNominalMeasures + DiscretizeByFrequency (bins=5) + WeightByDataToWeights/UserSpec
11	AutoMLP + NominalToNumerical + WeightByUserSpec
	NeuralNet + NominalToNumerical + NormalizeByZ/Range + WeightByDataToWeights/WeightByUserSpec/WeightByValueAverage
	NaiveBayesKernel + NominalToNumerical + WeightByUserSpec/DataToWeights/WeightByValueAverage
	kNNMixedMeasures + NominalToNumerical + NormalizeByZ + WeightByValueAverage
	kNNNumericalMeasures + NominalToNumerical + NormalizeByZ + WeightByDataToWeights
10	kNNMixedMeasures + NormalizeByRange/Z + WeightByUserSpec/DatatToWeights
	kNNMixedMeasures/NominalMeasures + DiscretizeByBinning (bins=10) + WeightBydataToWeights

Table C.3.: Good classification workflows for testing datasets on average

# Ds	Workflow description
26	kNNNominalMeasures + DiscretizeByBinning (bins=5,3) + WeightByInfoGainRatio/ByChiSquared/ByUncertainty kNNMixedMeasures + DiscretizeByBinning (bins=5,3)+ WeightByInfoGainRatio/ChiSquared/Uncertainty
25	kNNMixedMeasures + DiscretizeByBinning (bins=2) + WeightByInfoGain/WeightByUncertainty kNNNominalMeasures + DiscretizeByBinning (bins=2) + WeightByInfoGain/ByUncertainty
24	kNNNominalMeasures + DiscretizeByBinning/BySize (bins=5,10) + WeightByRule kNNMixedMeasures + NormalizeByRange/Z + WeightByUserSpec/DataToWeights
21	RegularizedDiscriminantAnalysis + NormalizeRange +WeightByUncertainty DefaultModel + NormalizeByRange/ByZ + WeightByValueAverage/WeightByDeviation/WeightByPCA/WeightBySVM
20	RegularizedDiscriminantAnalysis + NormalizeByRange/NoNorm + WeightByValue/DataToWeights/UserSpec
19	QuadraticDiscriminantAnalysis + NormalizeByRange + WeightBySVM/ByPCA
17	SingleRuleInductionSingleAttribute + DiscretizeByFrequency (bins=3) + WeightByUncertainty/InfoGain/ChiSquared/WeightByGiniIndex
16	kNNNominalMeasures + DiscretizeByFrequency (bins=3) + WeightByUncertainty/InfoGain/Ratio kNNMixedMeasures + DiscretizeByFrequency (bins=3) + WeightByUncertainty/InfoGain/Ratio
14	kNNNominalMeasures + DiscretizeByFrequency (bins=5) + WeightByUncertainty/InfoGain/Ratio kNNMixedMeasures + DiscretizeByFrequency (bins=5) + WeightByUncertainty/InfoGain/Ratio
11	kNNNumericalMeasures + NominalToNumerical + NormalizebyZ/Range + WeightByRelief/Uncertainty kNNMixedMeasures + NominalToNumerical + NormalizeByZ + WeightByUncertainty
10	RegularizedDiscriminantAnalysis + NominalToNumerical + NormalizeByRange + WeightByInfoGainRatio/UserSpec/DataToWeights/ValueAverage

Table C.4.: Worst performing classification workflows for testing datasets on average

Dataset	# wfs =	# wfs >	# wfs <
Annealing	[27,27,27,26]	[90,90,90,85	[19,19,19,25]]
CylinderBands	[24,24,20,27]	[94,80,88,78]	[12,26,22,25]
HeartH	[27,25,27,24]	[96,94,92,95]	[25,29,29,29]
Hepatitis	[31,31,28,28]	[74,68,70,68]	[48,45,46,39]
Mammographic	[18,17,18,17]	[102,101,98,97]	[19,21,23,25]
Soybean	[4,4,4,7]	[25,25,25,22]	[1,1,1,1]
Vote	[8,8,8,4]	[15,15,15,23]	[8,8,8,4]
HorseColic	[18,23,23,18]	[96,81,84,97]	[18,28,25,17]
Eucalyptus	[39,25,18,32]	[78,99,96,80]	[25,18,28,30]
Biomed	[48,44,31,46]	[73,71,88,79]	[53,59,55,49]

Table C.5.: Case-base datasets that perform well when using 'fill missing values'

C.2. Preprocessing

Pre-processing has shown to have a positive effect for certain workflows. In the following we analyze the datasets for which certain pre-processing methods impact positively or negatively the performance of workflows.

C.2.1. Fill Missing Values

The number of datasets for which the methods for filling missing values work better than without is comparable to the ones for which not filling missing values is better. Here, for many workflows and datasets it does not matter if we use the dataset as it it. Moreover, keeping the data with missing values may give better performance for more datasets than replacing them with statistics.

As the results are mixed and do not show clearly which method works better, we go one step further and analyse the performance of the workflows per dataset. We divide datasets in three categories: good performing datasets with fill missing values operators, bad performing datasets and no care datasets. The good performing datasets are shown in Table C.5. The number of good performing workflows is significantly higher than the ones for each the workflow perform equally or worse. There are other datasets for which filling missing values still works slightly better than keeping the original data (e.g., AustralianCreditApproval, BreastCancer, BreastCancerWDiagnostic, CreditApproval, Dermatology,etc.). But there are also a small set of datasets for which keeping the missing values brings better performance. These are shown

Dataset	# wfs =	# wfs >	# wfs <
HeartC	[36,53,53,33]	[44,49,49,39]	[70,48,48,78]]
HeartDisease	[36,53,53,33]	[44,49,49,39]	[70,48,48,78]
PrimaryTumor	[13,13,13,13]	[5,5,5,5]	[12,12,12,12]
BroadwayMult	[45,42,43,43]	[50,57,61,62]	[59,55,50,49]

Table C.6.: Case-base datasets that perform bad when using 'fill missing values'

Dataset	# wfs =	# wfs >	# wfs <
Adult	[47,47,47,45]	[41,41,41,51]	[25,25,25,17]
InternetAds	[67,74,74,70]	[68,85,85,78]	[12,7,7,9]
CJS	[44,45,48,69]	[67,59,58,39]	[27,26,24,22]
BankMarketing	[76,76,76,75]	[42,42,42,40]	[10,10,10,13]

Table C.7.: Test datasets that perform good when using 'fill missing values'

in table C.8. The question that we are asking is why are these specific datasets performing bad when filling in the missing values.

There are some datasets for which both methods work equally well (with minor differences: Arrhytmia, AustralianCreditApproval, Mushroom, HallOfFame, Secom, ...).

C.2.2. Normalization

As for the previous two pre-processing methods we compare the number of datasets and workflows per dataset.

To have a better idea we have also collected the statistics for good, bad and no care datasets. The selected datasets must have the number of workflows that perform better with a selected metric at least as twice than the number of workflows for which

Dataset	# wfs =	# wfs >	# wfs <
AdultKDD	[70,69,70,78]	[8,8,8,9]	[38,38,32,29]
ImageSegmentation	[14,14,14,14]	[68,85,85,78]	[107,107,107,107]
Ozone	[62,59,56,57]	[17,24,23,19]	[42,38,42,55]
Hypothyroid	[81,84,83,84]	[26,41,50,46]	[70,52,44,38]
Marketing	[12,12,12,12]	[3,3,3,7]	[15,15,15,11]

Table C.8.: Test datasets that perform bad when using 'fill missing values'

Dataset	# wfs $=$	# wfs $>$	# wfs $<$
BreastCancer	[521,612]	[227,185]	[96,38]
BreastTissue	[119,-]	[64,-]	[25,-]
IndianLiver	[806,841]	[259,228]	[115,111]
Haberman	[-,219]	[-,48]	[-,22]
HeartC	[710,-]	[340,-]	[130,-]
HeartDisease	[710,-]	[340,-]	[130,-]
HeartH	[677,-]	[342,-]	[149,-]
Lenses	[-,162]	[-,14]	[-,6]
PimaIndianDiabetes	[112,126]	[67,62]	[29,20]
StatlogHeart	[134,114]	[66,68]	[8,26]
Vote	[649,-]	[133,-]	[62,-]
Vowel	[169,-]	[81,-]	[39,-]
Wine	[122,-]	[58,-]	[28,-]
PittsburghBridges	[698,680]	[200,180]	[50,88]
BrazilTourism	[-,770]	[-,144]	[-,58]
BroadwayMult	[918,970]	[115,96]	[51,18]
Irish	[908,-]	[203,-]	[69,-]
Collins	[58,-]	[60,-]	[29,-]
prnn_crabs	[154,-]	[114,-]	[21,-]
Asbestos	[200,210]	[78,62]	[11,17]
cyyoung9302	[188,198]	[71,56]	[17,22]
DMFT	[128,144]	[52,37]	[12,11]

Table C.9.: Case-base datasets that perform good when using Z-transformation respective range transformation

the normalization metric performs worse compared to the default method. In the table we display the results obtained by both normalization methods. For some datasets normalize by z-transformation works better than range transformation (Indian Liver, PittsburghBridges, BroadwayMult, etc.).

We computed the same statistics for the test datasets in Table C.11 and Table C.12.

C.3. Comparison of selection strategies

As in the comparison section we have presented the results for the best strategies, here we give an overview of all the methods in Table 5.1. We present the results for

Dataset	# wfs $=$	# wfs $>$	# wfs $<$
Baloons	[164,-]	[9,-]	[22,-]
MolecularBiologyPromoters	[90,134]	[21,17]	[22,54]
ScaleBalance	[144,-]	[11,-]	[40,-]
TeachingAssistant	[-,197]	[-,27]	[-,65]
Trains	[197,213]	[16,8]	[63,56]
Zoo	[-,223]	[-,8]	[-,23]
Irish	[-,923]	[-,85]	[-,172]
Schizo	[830,776]	[109,93]	[241,311]
VertebralColumnC3	[-,118]	[-,23]	[-,67]
Cars	[-,790]	[-,110]	[-,228]
Cloud	[-,167]	[-,32]	[-,90]
Lupus	[-,128]	[-,21]	[-,59]
germangss	[-,186]	[-,33]	[-,70]

Table C.10.: Case-base datasets that perform bad when using Z-transformation respective range transformation

Dataset	# wfs $=$	# wfs $>$	# wfs $<$
Adult	[836,920]	[238,136]	[41,61]
ChessKRKPA7	[165,-]	[27,-]	[9,-]
JVowel	[93,-]	[72,-]	[17,-]
MFeatFourier	[111,111]	[70,83]	[24,10]
MFeatMorphological	[136,-]	[43,-]	[21,-]
PageBlocks	[107,-]	[61,-]	[27,-]
Yeast	[-,135]	[-,41]	[-,15]
Secom	[-,702]	[-,49]	[-,24]
SensorReadings24	[121,119]	[52,52]	[22,24]
SteelPlatesFaults	[170,-]	[75,-]	[26,-]

Table C.11.: Test datasets that perform good when using Z-transformation respective range transformation

Dataset	# wfs $=$	# wfs $>$	# wfs $<$
AdultKDD	[-,752]	[-,40]	[-,135]
MFeatKarhunen	[-,122]	[-,13]	[-,71]
MFeatPixel	[118,-]	[28,-]	[58,-]
Mushroom	[-,604]	[-,48]	[-,192]
Musk2	[-,147]	[-,15]	[-,33]
Nursery	[-,118]	[-,9]	[-,29]
PenDigits	[-,150]	[-,9]	[-,42]
WaveformV2	[-,121]	[-,21]	[-,66]
Hypothyroid	[-,696]	[-,84]	[-,192]
CJS	[-,747]	[-,84]	[-,245]
Marketing	[-,645]	[-,16]	[-,183]
RingNorm	[-,94]	[-,25]	[-,70]
TwoNorm	[-,116]	[-,30]	[-,62]

Table C.12.: Test datasets that perform bad when using Z-transformation respective range transformation

the kNN method. For selection we compare both offline and online strategies. For classification problems the best MAE results are obtained by the Random method followed closely by IGCN, GRCN and Pop. The results for MAE Loss varies with the number of workflows. For regression the results are similar Pop and the online strategies performing relatively better.

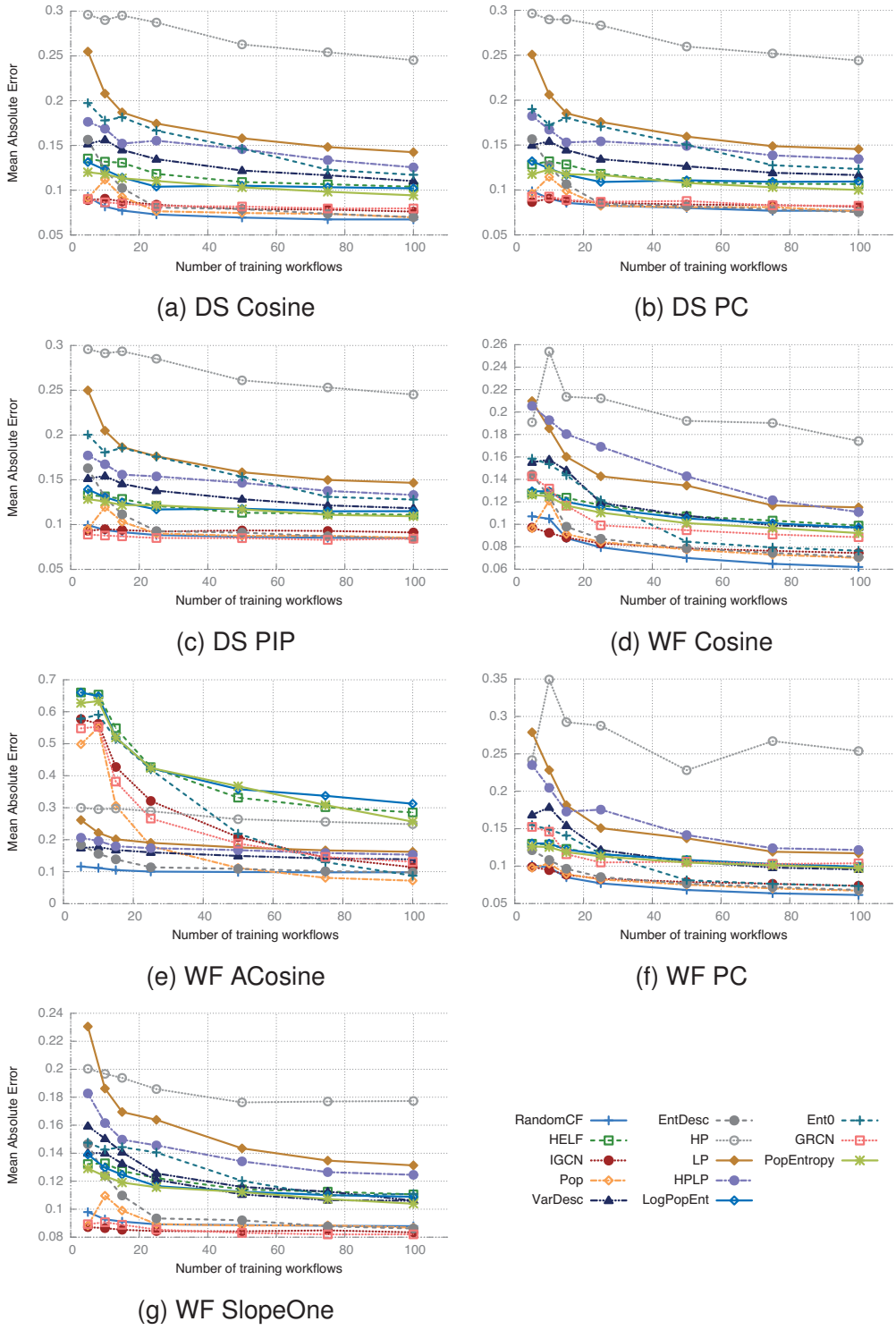


Figure C.1.: MAE for Dataset and Workflow-based CF (classification task)

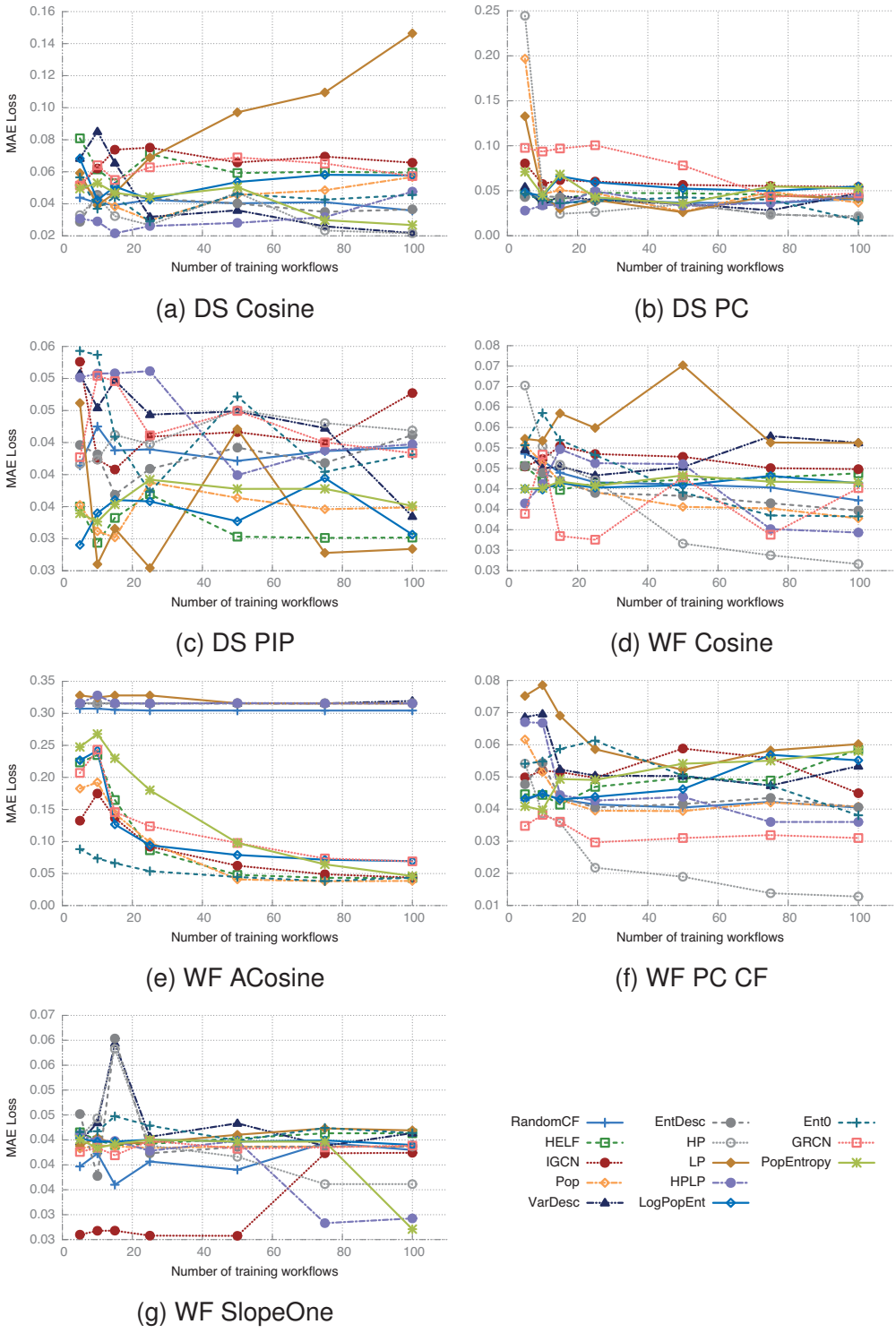
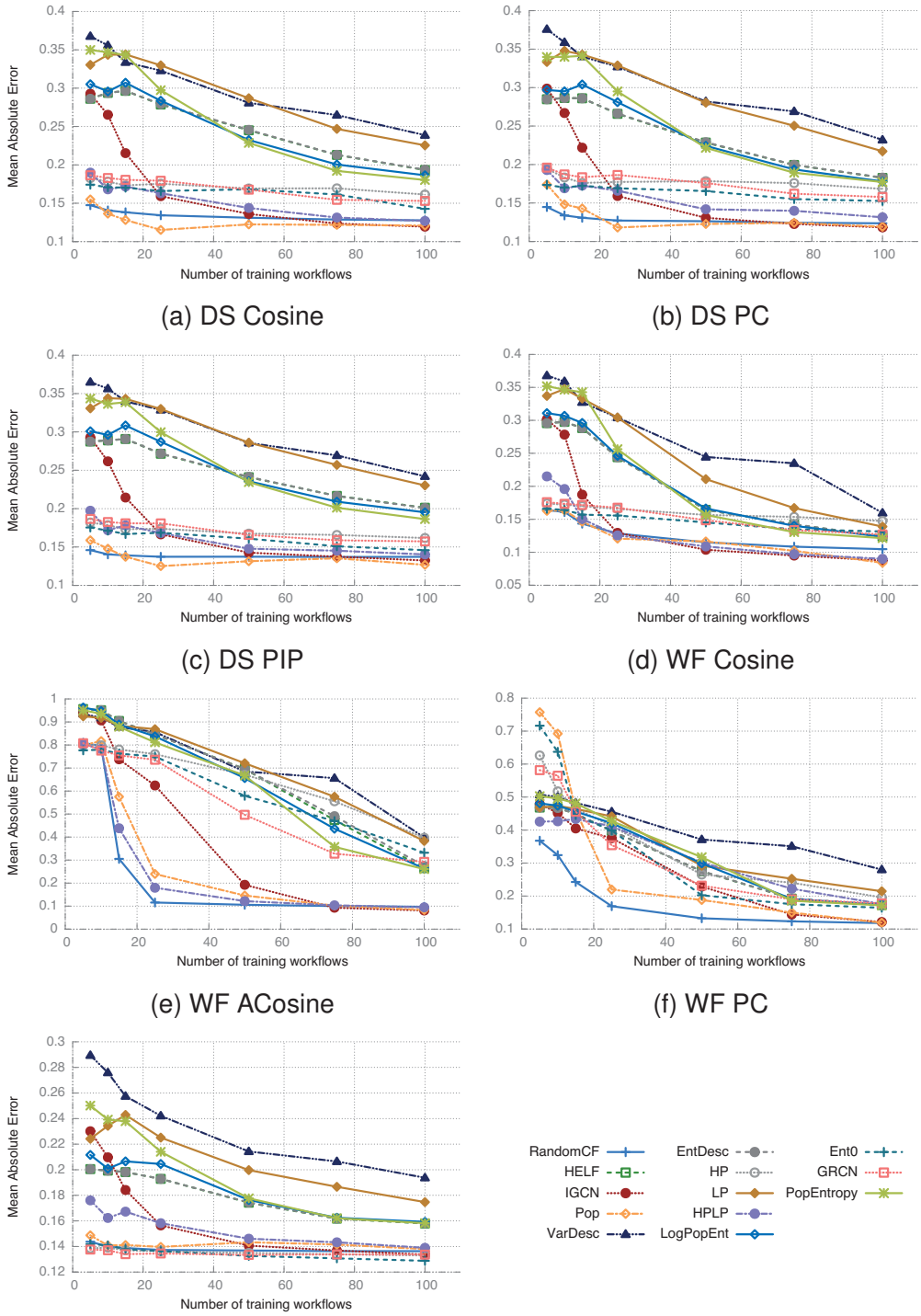


Figure C.2.: MAE Loss between Top10 predicted and best real workflow for Dataset and Workflow-based CF



(a) DS Cosine

(b) DS PC

(c) DS PIP

(d) WF Cosine

(e) WF ACosine

(f) WF PC

(g) WF SlopeOne

Figure C.3.: MAE for Dataset and Workflow-based CF (regression task)

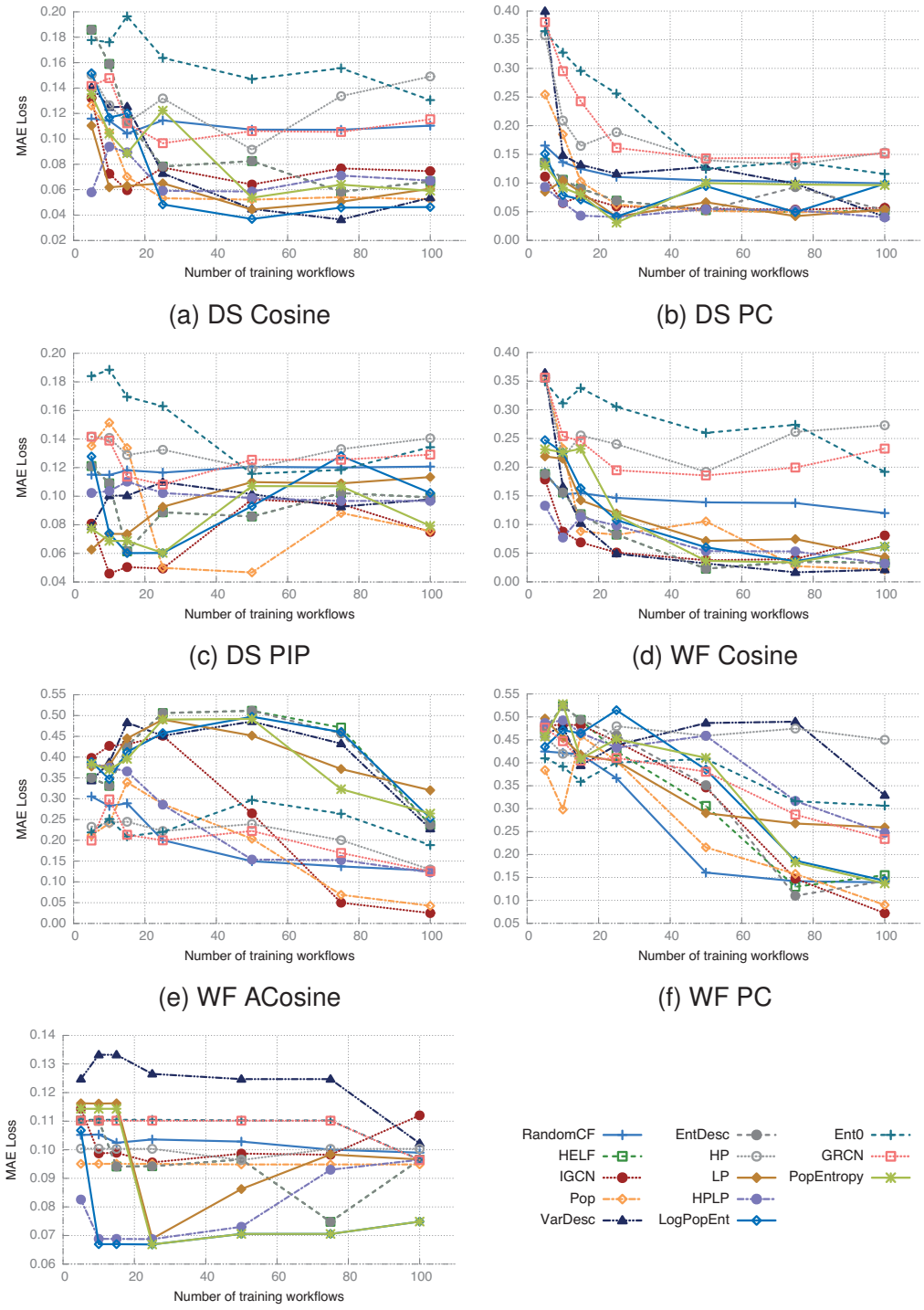


Figure C.4.: NAR Loss between Top5 predicted and best real workflow for Dataset and Workflow-based CF

Bibliography

[Aha et al., 1992] Aha, D. et al. (1992). Generalizing from case studies: A case study. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 1–10.

[Ahn, 2008] Ahn, H. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51.

[Amant and Cohen, 1995] Amant, R. and Cohen, P. (1995). Preliminary system design for an eda assistant. Technical report, University of Massachusetts, Amherst, MA, USA.

[Amant and Cohen, 1998a] Amant, R. and Cohen, P. (1998a). Interaction with a mixed-initiative system for exploratory data analysis. *Knowledge-Based Systems*, 10(5):265–273.

[Amant and Cohen, 1998b] Amant, R. S. and Cohen, P. (1998b). Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics*, 7(4):545–558.

[Anderson, 2010] Anderson, C. (2010). *The long tail: how endless choice is creating unlimited demand*. Cornerstone Digital.

[Azevedo, 2008] Azevedo, A. (2008). Kdd, semma and crisp-dm: a parallel overview.

[Bell et al., 2009] Bell, R., Bennett, J., Koren, Y., and Volinsky, C. (2009). The million dollar programming prize. *Spectrum, IEEE*, 46(5):28–33.

[Bell and Koren, 2007] Bell, R. M. and Koren, Y. (2007). Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79.

[Bensusan and Kalousis, 2001] Bensusan, H. and Kalousis, A. (2001). Estimating the predictive accuracy of a classifier. *Lecture Notes in Computer Science*, 2167:25–36.

[Berka and Bruha, 1998] Berka, P. and Bruha, I. (1998). Discretization and grouping: Preprocessing steps for data mining. *Principles of Data Mining and Knowledge Discovery*, pages 239–245.

[Bernstein and Daenzer, 2007] Bernstein, A. and Daenzer, M. (2007). The NExT system: Towards true dynamic adaptations of semantic web service compositions. *Lecture Notes in Computer Science*, 4519:739–748.

[Bernstein et al., 2005] Bernstein, A., Provost, F., and Hill, S. (2005). Toward intelligent assistance for a data mining process: an ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):503–518.

[Berthold et al., 2009] Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., and Wiswedel, B. (2009). Knime - the konstanz information miner: version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11:26–31.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

[Blum and Furst, 1997] Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis* 1. *Artificial intelligence*, 90(1-2):281–300.

[Botia et al., 2001] Botia, J., Gomez-Skarmeta, A., Valdes, M., and Padilla, A. (2001). METALA: A meta-learning architecture. *Lecture Notes in Computer Science*, 2206:688–698.

[Boutilier et al., 2010] Boutilier, C., Regan, K., and Viappiani, P. (2010). Simultaneous elicitation of preference features and utility. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1160–1197.

[Brachman and Anand, 1996] Brachman, R. and Anand, T. (1996). The process of knowledge discovery in databases. In *Advances in knowledge discovery and data mining*, pages 37–57. American Association for Artificial Intelligence.

[Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, 461(8).

[Cannataro and Comito, 2003] Cannataro, M. and Comito, C. (2003). A data mining ontology for grid programming. *Proceedings of (SemPGrid2003)*, pages 113–134.

[Castiello et al., 2005] Castiello, C., Castellano, G., and Fanelli, A. (2005). Meta-data: Characterization of input features for meta-learning. *Modeling Decisions for Artificial Intelligence*, pages 295–304.

[Castiello and Fanelli, 2005] Castiello, C. and Fanelli, A. (2005). Meta-learning experiences with the mindful system. *Lecture Notes in Artificial Intelligence*, 3801:321–328.

[Cerrito, 2007] Cerrito, P. (2007). *Introduction to Data Mining Using SAS Enterprise Miner*. SAS Publishing, NC, USA.

[Chandrasekaran et al., 1992] Chandrasekaran, B., Johnson, T., and Smith, J. (1992). Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35(9):124–137.

[Chapman et al., 1999] Chapman, P., Clinton, J., Khabaza, T., Reinartz, T., and Wirth, R. (1999). The crisp-dm process model. *The CRISP–DM Consortium*, 310.

[Charest et al., 2008] Charest, M., Delisle, S., Cervantes, O., and Shen, Y. (2008). Bridging the gap between data mining and decision support: A case-based reasoning and ontology approach. *Intelligent Data Analysis*, 12:1–26.

[Choinski and Chudziak, 2009] Choinski, M. and Chudziak, J. (2009). Ontological learning assistant for knowledge discovery and data mining. In *Computer Science and Information Technology*, pages 147–155. IEEE.

[Cios et al., 2010] Cios, K., Pedrycz, W., Swiniarski, R., and Kurgan, L. (2010). *Data mining: a knowledge discovery approach*. Springer Publishing Company, Incorporated.

[Craw et al., 1992] Craw, S., Sleeman, D., Graner, N., and Rissakis, M. (1992). Consultant: Providing advice for the machine learning toolbox. In *Proceedings of the Annual Technical Conference on Expert Systems (ES)*, pages 5–23.

[Crone et al., 2006] Crone, S., Lessmann, S., and Stahlbock, R. (2006). The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research*, 173(3):781–800.

[Das et al., 2007] Das, A., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM.

[Dean and Ghemawat, 2008] Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

[Demšar et al., 2004] Demšar, J., Zupan, B., Leban, G., and Curk, T. (2004). Orange: From experimental machine learning to interactive data mining. In *Knowledge Discovery in Databases: PKDD 2004*, pages 537–539, London, UK. Springer.

[Diamantini et al., 2009a] Diamantini, C., Potena, D., and Storti, E. (2009a). Kddonto: An ontology for discovery and composition of kdd algorithms. pages 13–24.

[Diamantini et al., 2009b] Diamantini, C., Potena, D., and Storti, E. (2009b). Ontology-driven KDD process composition. *Lecture Notes in Computer Science*, 5772:285–296.

[Dougherty et al., 1995] Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 194–202. Morgan Kaufmann Publishers, Inc.

[Džeroski, 2007] Džeroski, S. (2007). Towards a general framework for data mining. *Knowledge Discovery in Inductive Databases*, pages 259–300.

[Elahi et al., 2011] Elahi, M., Repsys, V., and Ricci, F. (2011). Rating elicitation strategies for collaborative filtering. *E-Commerce and Web Technologies*, pages 160–171.

[Engels, 1996] Engels, R. (1996). Planning tasks for knowledge discovery in databases: Performing task-oriented user-guidance. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD)*, pages 170–175.

[Engels et al., 1997] Engels, R., Lindner, G., and Studer, R. (1997). A guided tour through the data mining jungle. In *Proceedings of the 3nd International Conference*

on Knowledge Discovery in Databases, pages 163–166, Menlo Park, CA, USA. AAAI Press.

[Erol, 1996] Erol, K. (1996). *Hierarchical task network planning: formalization, analysis, and implementation*. PhD thesis, University of Maryland at College Park, College Park, MD, USA. UMI Order No. GAX96-22054.

[Erol et al., 1995] Erol, K., Hendler, J., and Nau, D. (1995). HTN planning: Complexity and expressivity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1123–1123. JOHN WILEY & SONS LTD.

[Fayyad et al., 1996a] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

[Fayyad et al., 1996b] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996b). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

[Fennell, 2009] Fennell, J. (2009). Collaborative filtering on sparse rating data for yelp. com.

[Frank and Asuncion, 2010] Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

[Frawley et al., 1992] Frawley, W., Piatetsky-Shapiro, G., and Matheus, C. (1992). Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57.

[Fürnkranz and Petrak, 2001] Fürnkranz, J. and Petrak, J. (2001). An evaluation of landmarking variants. In *Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001)*, pages 57–68.

[Gale, 1986] Gale, W. (1986). Rex review. In *Artificial intelligence and statistics*, pages 173–227, Boston, Massachusetts. Addison-Wesley Longman Publishing Co., Inc.

[Gama and Brazdil, 1995] Gama, J. and Brazdil, P. (1995). Characterization of classification algorithms. *Progress in Artificial Intelligence*, pages 189–200.

[Giraud-Carrier, 2005] Giraud-Carrier, C. (2005). The data mining advisor: meta-learning at the service of practitioners. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, pages 113–119. IEEE Computer Society.

[Grabczewski and Jankowski, 2007] Grabczewski, K. and Jankowski, N. (2007). Versatile and efficient meta-learning architecture: Knowledge representation and .... *IEEE Symposium on Computational Intelligence and Data Mining*, pages 51–58.

[Graner et al., 1993] Graner, N., Sharma, S., Sleeman, D., Rissakis, M., CRAW, S., and MOORE, C. (1993). The machine learning toolbox consultant. *International Journal on AI Tools*, 2(3):307–328.

[Grimmer, 1996] Grimmer, U. (1996). Clementine: Data mining software. *Classification and Multivariate Graphics: Models, Software and Applications*, pages 25–31.

[Gruber, 2008] Gruber, T. (2008). What is an ontology. *Encyclopedia of Database Systems*, 1.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[Hand, 1985] Hand, D. (1985). Statistical expert systems: necessary attributes. *Journal of applied Statistics*, 12(1):19–27.

[Hand, 1987] Hand, D. (1987). A statistical knowledge enhancement system. *Journal of the Royal Statistical Society. Series A (General)*, 150(4):334–345.

[Hand, 1990] Hand, D. (1990). Practical experience in developing statistical knowledge enhancement systems. *Annals of Mathematics and Artificial Intelligence*, 2(1):197–208.

[Hannon et al., 2010] Hannon, J., Bennett, M., and Smyth, B. (2010). Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM.

[Harpale and Yang, 2008] Harpale, A. S. and Yang, Y. (2008). Personalized active learning for collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–98. ACM.

[Harris Jr, 2002] Harris Jr, E. (2002). Information gain versus gain ratio: A study of split method biases. *AMAI*.

[Hearst, 1999] Hearst, M. (1999). Mixed-initiative interaction. *IEEE Intelligent systems*, 14(5):14–23.

[Heckert and Filliben, 2003] Heckert, A. and Filliben, J. (2003). Dataplot reference manual. 148.

[Hernansaez et al., 2004] Hernansaez, J., Botıa, J., and Skarmeta, A. (2004). METALA: a J2EE technology based framework for web mining. *Revista Colombiana de Computación*, 5(1).

[Hilario et al., 2011] Hilario, M., Nguyen, P., Do, H., Woznica, A., and Kalousis, A. (2011). Ontology-based meta-mining of knowledge discovery workflows. *Meta-Learning in Computational Intelligence*, pages 273–315.

[Hoffmann and Nebel, 2001] Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

[Hoos, 2012] Hoos, H. H. (2012). Programming by optimization. *Communications of the ACM*, 55(2):70–80.

[Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P., and Boley, H. (2004). SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21(http://www.w3.org/Submissions/SWRL/).

[Horvitz, 1999] Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pages 159–166. ACM.

[Hutter et al., 2012] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2012). Identifying key algorithm parameters and instance features using forward selection. In *Learning and Intelligent Optimization (LION 7)*.

[Ihaka and Gentleman, 1996] Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314.

[Kalousis, 2002] Kalousis, A. (2002). *Algorithm selection via meta-learning*. PhD thesis.

[Kalousis and Hilario, 2001a] Kalousis, A. and Hilario, M. (2001a). Feature selection for meta-learning. *Advances in Knowledge Discovery and Data Mining*, 2035:222–233.

[Kalousis and Hilario, 2001b] Kalousis, A. and Hilario, M. (2001b). Model selection via meta-learning: a comparative study. *International Journal on Artificial Intelligence Tools*, 10(4):525–554.

[Kalousis and Hilario, 2003] Kalousis, A. and Hilario, M. (2003). Representational issues in meta-learning. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 313.

[Kalousis and Theoharis, 1999] Kalousis, A. and Theoharis, T. (1999). Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(4):319–337.

[Kietz et al., 2010a] Kietz, J. U., Serban, F., and Bernstein, A. (2010a). eProPlan: a tool to model automatic generation of data mining workflows. In *3rd Planning to Learn Workshop (WS9) at ECAI'10*, pages 15–17.

[Kietz et al., 2009] Kietz, J.-U., Serban, F., Bernstein, A., and Fischer, S. (2009). Towards cooperative planning of data mining workflows. In *Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-09)*.

[Kietz et al., 2010b] Kietz, J.-U., Serban, F., Bernstein, A., and Fischer, S. (2010b). Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In *Proc of the ECML/PKDD'10 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD'10)*, pages 1–12.

[Kietz et al., 2012] Kietz, J.-U., Serban, F., Bernstein, A., and Fischer, S. (2012). Designing KDD-Workflows via HTN-Planning for Intelligent Discovery Assistance. In

Vanschoren, J., Kietz, J.-U., and Brazdil, P., editors, *Planning to Learn 2012, Workshop at ECAI 2012*, CEUR Workshop Proceedings.

[Kodratoff et al., 1992] Kodratoff, Y., Sleeman, D., Uszynski, M., Causse, K., and Craw, S. (1992). Building a machine learning toolbox. *In: Enhancing the knowledge engineering process: contributions from ESPRIT (eds. L Steels and B Lepape). Elsevier*, pages 81–108.

[Kohavi et al., 2000] Kohavi, R., Brodley, C. E., Frasca, B., Mason, L., and Zheng, Z. (2000). Kdd-cup 2000 organizers' report: peeling the onion. *SIGKDD Explor. Newsl.*, 2:86–93.

[Kohrs and Merialdo, 2001] Kohrs, A. and Merialdo, B. (2001). Improving collaborative filtering for new users by smart object selection. In *ICMF'01*.

[Köpf et al., 2000] Köpf, C., Taylor, C., and Keller, J. (2000). Meta-analysis: From data characterisation for meta-learning to meta-regression. In *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support,Meta-Learning and ILP*.

[Koren, 2009] Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[Kriegel et al., 2007] Kriegel, H., Borgwardt, K., Kröger, P., Pryakhin, A., Schubert, M., and Zimek, A. (2007). Future trends in data mining. *Data Mining and Knowledge Discovery*, 15(1):87–97.

[Lam et al., 2008] Lam, X. N., Vu, T., Le, T. D., and Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211. ACM.

[LaValle et al., 2011] LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., and Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. *MIT sloan management review*, 52(2):21–32.

[Leite and Brazdil, 2010] Leite, R. and Brazdil, P. (2010). Active testing strategy to predict the best classification algorithm via sampling and metalearning. In *Proceedings of the 2010 conference on ECAI*, pages 309–314.

[Lemire and Maclachlan, 2005] Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. *Society for Industrial Mathematics*, 5:471–480.

[Levesque, 2005] Levesque, R. (2005). *SPSS programming and data management: A guide for SPSS and SAS users*. SPSS, Chicago, USA.

[Levine et al., 1999] Levine, D., Berenson, M., and Stephan, D. (1999). *Statistics for managers using Microsoft Excel*. Prentice Hall, New Jersey, USA.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80.

[Lindner and Studer, 1999a] Lindner, G. and Studer, R. (1999a). Ast: Support for algorithm selection with a cbr approach. *Principles of Data Mining and Knowledge Discovery*, pages 418–423.

[Lindner and Studer, 1999b] Lindner, G. and Studer, R. (1999b). AST: Support for algorithm selection with a CBR approach. *Lecture Notes in Computer Science*, 1704:418–423.

[Liu and Lee, 2010] Liu, F. and Lee, H. (2010). Use of social network information to enhance collaborative filtering performance. *Expert Systems with Applications*, 37(7):4772–4778.

[Lohr, 2012] Lohr, S. (2012). The age of big data. *New York Times*, 11.

[Ma, 2008] Ma, C.-C. (2008). A guide to singular value decomposition for collaborative filtering.

[Manyika et al., 2011] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, pages 1–137.

[Marlin, 2003] Marlin, B. (2003). Modeling user rating profiles for collaborative filtering. *Advances in neural information processing systems*, 16.

[Marlin, 2004] Marlin, B. (2004). *Collaborative filtering: A machine learning perspective*. PhD thesis, University of Toronto.

[MathWorks, 2004] MathWorks, I. (2004). Matlab. *The MathWorks, Natick, MA*.

[McCuinness, 2005] McCuinness, D. (2005). Ontologies come of age. *Spinning the semantic web: bringing the World Wide Web to its full potential*, page 171.

[McFee et al., 2012] McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R. (2012). The million song dataset challenge. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 909–916. ACM.

[Melville and Sindhwani, 2010] Melville, P. and Sindhwani, V. (2010). Recommender systems. *Encyclopedia of Machine Learning*, pages 829–838.

[Michie et al., 1994] Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA.

[Mierswa et al., 2006] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T. (2006). Yale: Rapid prototyping for complex data mining tasks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA. ACM.

[Mitchell, 1997] Mitchell, T. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*.

[Morik and Scholz, 2004] Morik, K. and Scholz, M. (2004). The MiningMart approach to knowledge discovery in databases. *In: Intelligent Technologies for Information Analysis (eds. N. Zhong, J. Liu), Springer*, pages 47–65.

[Myers, 1996] Myers, K. (1996). Strategic advice for hierarchical planners. In *Principles of knowledge representation and reasoning-International Conference-*, pages 112–123. Morgan Kaufmann Publishers.

[Nau et al., 2003] Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *J. Artif. Intell. Res. (JAIR)*, 20:379–404.

[Nau et al., 2004] Nau, D., Ghallab, M., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann, San Francisco, CA, USA.

[Nau et al., 1998] Nau, D. S., Smith, S. J. J., and Erol, K. (1998). Control strategies in htn planning: Theory versus practice. In *IAAI Proceedings*, pages 1127–1133.

[Newell et al., 1958] Newell, A., Shaw, J., and Simon, H. (1958). Elements of a theory of human problem solving. *Psychological Review*, 65(3):151.

[Nguyen et al., 2012] Nguyen, P., Wang, J., Hilario, M., and Kalousis, A. (2012). Learning heterogeneous similarity measures for hybrid-recommendations in meta-mining. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1026–1031. IEEE.

[OConnor and Herlocker, 1999] OConnor, M. and Herlocker, J. (1999). Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, pages 121–128. UC Berkeley.

[O'Mahony et al., 2008] O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., and O'Sullivan, B. (2008). Using case-based reasoning in an algorithm portfolio for constraint solving. In *Irish Conference on Artificial Intelligence and Cognitive Science*.

[Panov et al., 2008] Panov, P., Džeroski, S., and Soldatova, L. (2008). Ontodm: An ontology of data mining. In *Proceedings of the ICDM'08 Workshops*, pages 752–760.

[Patel-Schneider et al., 2004] Patel-Schneider, P., Hayes, P., Horrocks, I., et al. (2004). OWL web ontology language semantics and abstract syntax. *W3C recommendation*, 10.

[Pfahringer et al., 2000] Pfahringer, B., Bensusan, H., and Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. *Proceedings of the International Conference on Machine Learning (ICML)*, 951(2000):743–750.

[Prekopcsák et al., 2011] Prekopcsák, Z., Makrai, G., Henk, T., and Gáspár-Papanek, C. (2011). Radoop: Analyzing big data with rapidminer and hadoop. In *Proceedings of the 2nd RapidMiner Community Meeting and Conference (RCOMM 2011)*.

[Pyle, 1999] Pyle, D. (1999). *Data preparation for data mining*, volume 1. Morgan Kaufmann.

[Raes, 1992] Raes, J. (1992). Inside two commercially available statistical expert systems. *Statistics and Computing*, 2(2):55–62.

[Rashid et al., 2002] Rashid, A., Albert, I., Cosley, D., Lam, S., McNee, S., Konstan, J., and Riedl, J. (2002). Getting to know you: learning new user preferences in recommender systems. In *IUI'02*, pages 127–134. ACM.

[Rashid et al., 2008] Rashid, A., Karypis, G., and Riedl, J. (2008). Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100.

[Rennie and Srebro, 2005] Rennie, J. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM.

[Reuters, 2012] Reuters, T. (2012). Big data - graphic of the day.

[Rice, 1975] Rice, J. (1975). The algorithm selection problem.

[Rice, 1976] Rice, J. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–118.

[Rubens et al., 2011] Rubens, N., Kaplan, D., and Sugiyama, M. (2011). Active learning in recommender systems. In *Recommender Systems Handbook*, pages 735–767. Springer.

[Russell et al., 1993] Russell, D. M., Stefik, M. J., Pirolli, P., and Card, S. K. (1993). The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 269–276, New York, NY, USA. ACM.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

[Schafer et al., 2001] Schafer, J., Konstan, J., and Riedl, J. (2001). E-commerce recommendation applications. *Data mining and knowledge discovery*, 5(1):115–153.

[Schafer et al., 1999] Schafer, J. B., Konstan, J., and Riedi, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM.

[Schaffer, 1994] Schaffer, C. (1994). A conservation law for generalization performance. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 259–265.

[Schein et al., 2002] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the*

25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 253–260. ACM.

[Serban, 2010] Serban, F. (2010). Auto-experimentation of KDD workflows based on ontological planning. In *The 9th International Semantic Web Conference (ISWC 2010), Doctoral Consortium*, pages 313–320.

[Serban et al., 2012] Serban, F., Vanschoren, J., Kietz, J.-U., and Bernstein, A. (2012). A survey of intelligent assistants for data analysis. *ACM Computing Surveys*, forthcoming:1–35.

[Sirin and Parsia, 2007] Sirin, E. and Parsia, B. (2007). SPARQL-DL: SPARQL query for OWL-DL. In *Proceedings of the International Workshop on OWL Experiences and Directions (OWLED)*.

[Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53.

[Sirin et al., 2004] Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D. (2004). Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396.

[Sleeman et al., 1995] Sleeman, D., Rissakis, M., Craw, S., Graner, N., and Sharma, S. (1995). Consultant-2: Pre-and post-processing of machine learning applications. *International Journal of Human Computer Studies*, 43(1):43–63.

[Smith-Miles, 2008] Smith-Miles, K. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):Article 6.

[Soldatova and King, 2006] Soldatova, L. and King, R. (2006). An ontology of scientific experiments. *Journal of the Royal Society Interface*, 3(11):795–803.

[Srebro et al., 2004] Srebro, N., Rennie, J. D., and Jaakkola, T. (2004). Maximum-margin matrix factorization. In *NIPS*, volume 17, pages 1329–1336.

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4.

[Theus, 1998] Theus, M. (1998). Exploratory data analysis with data desk. *Computational Statistics*, 13(1):101–116.

[Todorovski et al., 2002] Todorovski, L., Blockeel, H., and Džeroski, S. (2002). Ranking with predictive clustering trees. *Lecture Notes in Computer Science*, 2430:444–455.

[Vanschoren, 2010] Vanschoren, J. (2010). *Understanding machine learning performance with experiment databases*. PhD thesis, Katholieke Universiteit Leuven.

[Vanschoren et al., 2012] Vanschoren, J., Blockeel, H., Pfahringer, B., and Holmes, G. (2012). Experiment databases: A new way to share, organize and learn from experiments. *Machine Learning*, (In press, DOI 10.1007/s10994-011-5277-0).

[Vilalta and Drissi, 2002] Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artif. Intell. Rev.*, 18:77–95.

[Weimer et al., 2007] Weimer, M., Karatzoglou, A., Le, Q., Smola, A., et al. (2007). Cofirank-maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems 20 21st Annual Conference on Neural Information Processing Systems 2007*, pages 222–230.

[Wirth et al., 1997a] Wirth, R., Shearer, C., Grimmer, U., Reinartz, T., Schlösser, J., Breitner, C., Engels, R., and Lindner, G. (1997a). Towards process-oriented tool support for knowledge discovery in databases. *Principles of Data Mining and Knowledge Discovery*, pages 243–253.

[Wirth et al., 1997b] Wirth, R., Shearer, C., Grimmer, U., Reinartz, T., Schlosser, J., Breitner, C., Engels, R., and Lindner, G. (1997b). Towards process-oriented tool support for knowledge discovery in databases. *Lecture Notes in Computer Science*, 1263:243–253.

[Witten and Frank, 2005] Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[Wolpert, 2001] Wolpert, D. (2001). The supervised learning no-free-lunch theorems. In *Proceedings of the Online World Conference on Soft Computing in Industrial Applications*, pages 25–42.

[Xu et al., 2008] Xu, L., Hutter, F., Hoos, H., and Leyton-Brown, K. (2008). Satzilla: portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research*, 32(1):565–606.

[Yang et al., 2003] Yang, G., Kifer, M., and Zhao, C. (2003). Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE)*.

[Yu-hua et al., 2006] Yu-hua, L., Zheng-ding, L., Xiao-lin, S., Kun-mei, W., and Rui-xuan, L. (2006). Data mining ontology development for high user usability. *Wuhan University Journal of Natural Sciences*, 11(1):51–56.

[Žáková et al., 2010] Žáková, M., Křemen, P., Železný, F., and Lavrač, N. (2010). Automatic knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering*, online 1st:53–264.

[Zhang et al., 2003] Zhang, S., Zhang, C., and Yang, Q. (2003). Data preparation for data mining. *Applied Artificial Intelligence*, 17(5-6):375–381.

Curriculum Vitae

Personal Details

Name	Floarea Serban
Place of Birth	Cluj-Napoca, Romania
Citizenship	Romanian

Education

2009-2013	*Doctor of Science*
	Department of Informatics, University of Zurich, Switzerland
2003-2008	*Dipl. Ing.*,
	Technical University of Cluj-Napoca, Romania

Relevant Research Experience

Dec. 2008-May 2013	*Research Assistant* Dynamic and Distributed Information Systems Group, Department of Informatics, University of Zurich
Sep.-Dec. 2008	Internship in the Systems Group, ETH Zurich
Jul.-Oct. 2007	Internship in the Computational Biophysics Lab, ETH Zurich
Apr.-Dec. 2007	Researcher at the Technical University of Cluj-Napoca